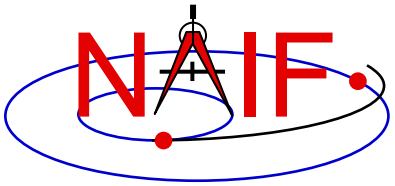
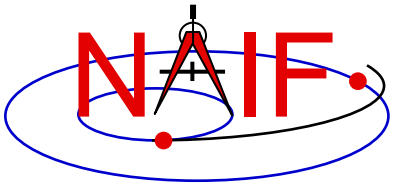


Navigation and Ancillary Information Facility

SPICE Geometry Finder (GF) Subsystem

Searching for times when specified geometric conditions occur

October 2022



Topics

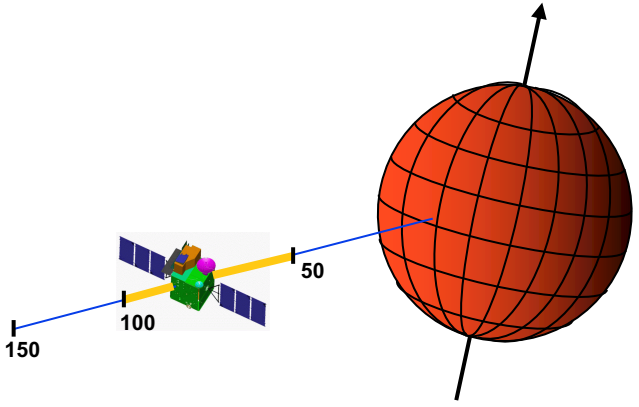
- GF Subsystem Overview
- SPICE Windows
- GF Search Examples
- Geometric Search Types and Constraints
- More Details
  - Root finding, including step size
  - Workspace
- An Example

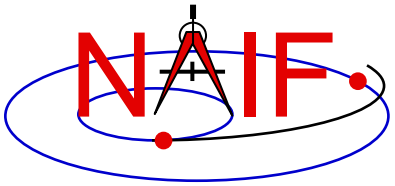
GF Subsystem Overview

Purpose

- **Much SPICE software computes a geometry parameter at a given time, t, i.e. *x = f(t)*.**
  - **Example: on 2011 MAR 30 14:57:08, what is the spacecraft's altitude above Mars?**

- **The Geometry Finder subsystem does the inverse: it finds times when specified geometric events occur.**
  - **Example: within some time bounds, when is the spacecraft's altitude between 50 and 100 km?**





Some Examples

- **The SPICE Geometry Finder (GF) subsystem finds times when specified geometric events occur.**
  - **A "geometric event" is an occurrence of a given geometric quantity satisfying a specified condition. For example:**
    - » **Mars Express distance from Mars is at a local minimum (periapse)**
    - » **Elevation of the Cassini orbiter is above a given threshold angle as seen from DSS-14**
    - » **Titan is completely occulted by Saturn**
    - » **The Saturn phase angle as seen by the Cassini orbiter is 60 degrees**
  - **Each GF search is conducted over a user-specified time window, called the confinement window.**
    - » **A "time window" is a union of time intervals.**
  - **The result of a GF search is the time window over which the specified condition is met.**

Types of GF APIs

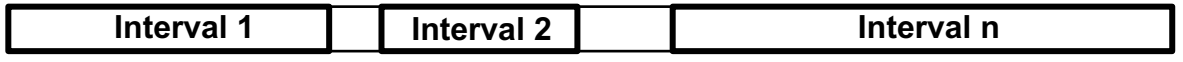
- GF provides two primary types of event-finding APIs
  - Boolean: a geometric condition (an event) is true or false
    - » Example: Phobos is occulted by Mars
    - » Example: Vesta is not in the OSIRIS instrument's field of view
  - We also call these binary conditions
  - Numeric: a geometric quantity has a given value, is within a given range or has achieved a local or global maximum or minimum
    - » Example: spacecraft altitude is between X and Y km above the surface
    - » Example: angular separation of Titan from Saturn has reached the maximum value

GF High-Level API Routines

- **The GF subsystem provides the following high-level API routines; these search for events involving the respective geometric quantities listed below**
  - **GFDIST: observer-target distance**
  - **GFILUM: illumination angles**
  - **GFOCLT: occultations or transits**
  - **GFPA: phase angle**
  - **GFPOSC: position vector coordinates**
  - **GFRFOV: ray is contained in an instrument's field of view**
  - **GFRR: observer-target range rate**
  - **GFSEP: target body angular separation**
  - **GFSNTC: ray-body surface intercept coordinates**
  - **GFSUBC: sub-observer point coordinates**
  - **GFTFOV: target body appears in an instrument's field of view**
  - **GFUDB: user-defined Boolean quantity (only Fortran, C and JNI)**
  - **GFUDS: user-defined scalar quantity (only Fortran, C and JNI)**

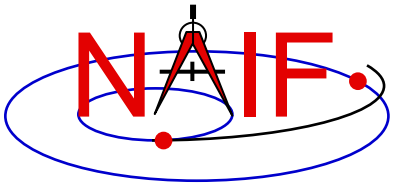
The SPICE Window

- The high-level GF routines return a search result as a SPICE window. This window specifies intervals of time when the user's constraints are satisfied.
- In simple terms, one can describe a SPICE window as:
  - A span of time defined by a start time and an end time, containing a list of disjoint intervals arranged in ascending order.
  - Within that time span, a time-ordered sequence of zero or more time intervals each having zero or non-zero length
    - » An interval is specified by a pair of double precision numbers, with the second greater than or equal to the first.
    - » A zero-length interval is often called a "singleton"



T_{start} T_{stop}

A SPICE Window



SPICE Windows Operations

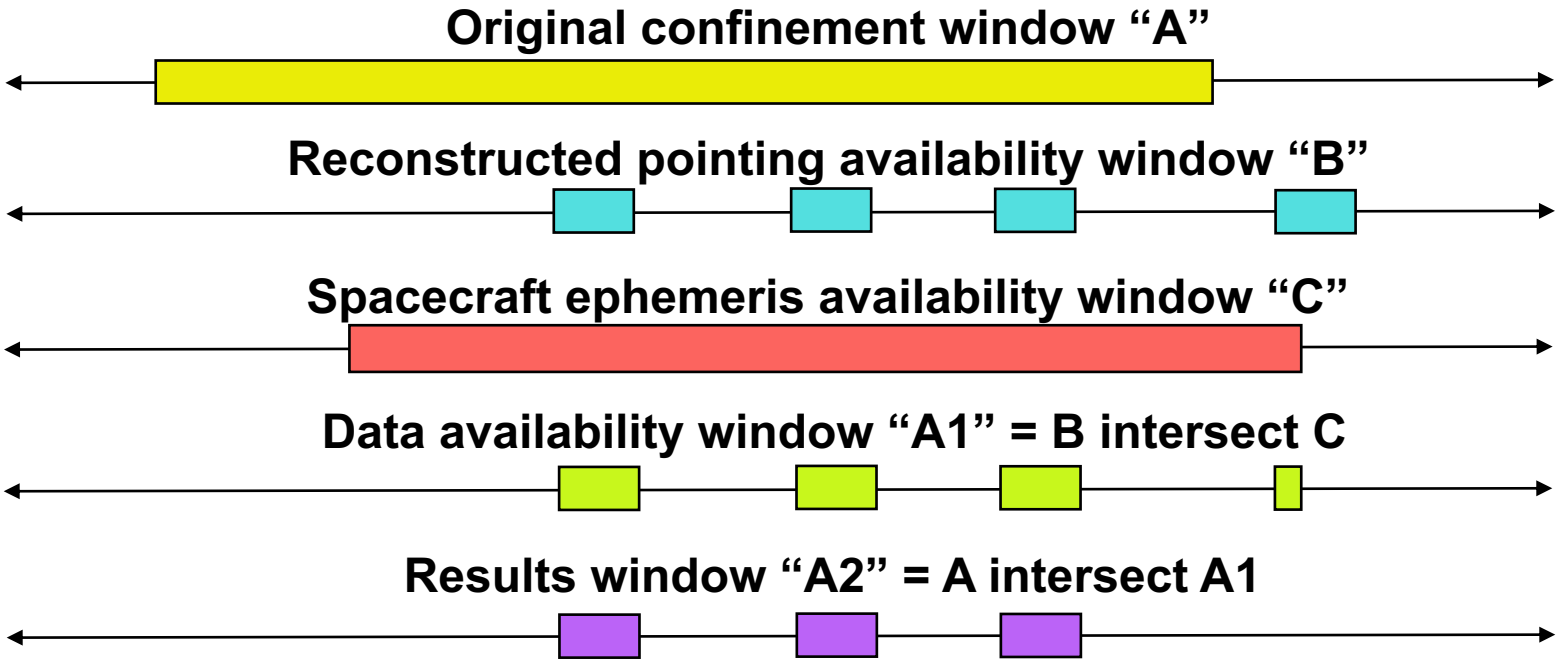
- **SPICE provides routines to:**
  - **compute unions, intersections, and differences of windows**
  - **contract each interval within a window …**
    - » **by increasing the left endpoint and decreasing the right endpoint**
- **These functions allow one to search for multi-condition events**
- **See the next page for an example**

Example of Window Operations

Given an initial confinement window, A, determine times when CK and SPK data are available within it.

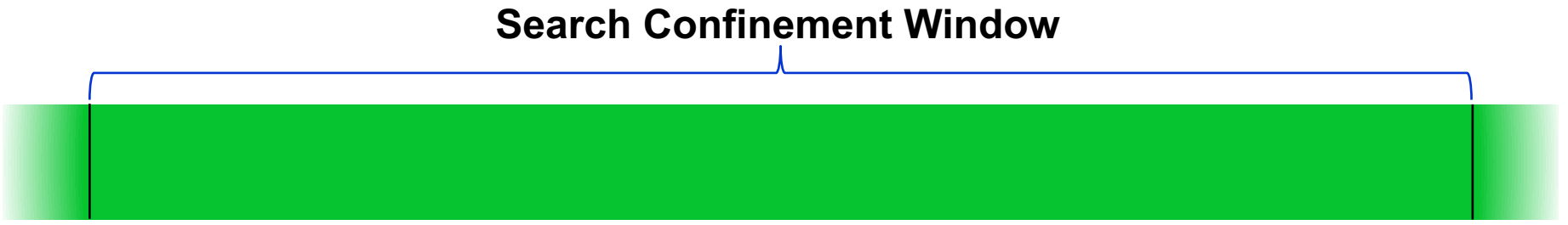
Use CKCOV and SPKCOV to find CK and SPK availability windows, B and C. Use window intersection to obtain the results window.

Contract the CK window slightly to avoid round-off problems. Contract the SPK window by a few seconds if discrete differentiation is used by search algorithms (e.g. for acceleration or for "is function decreasing?" tests).



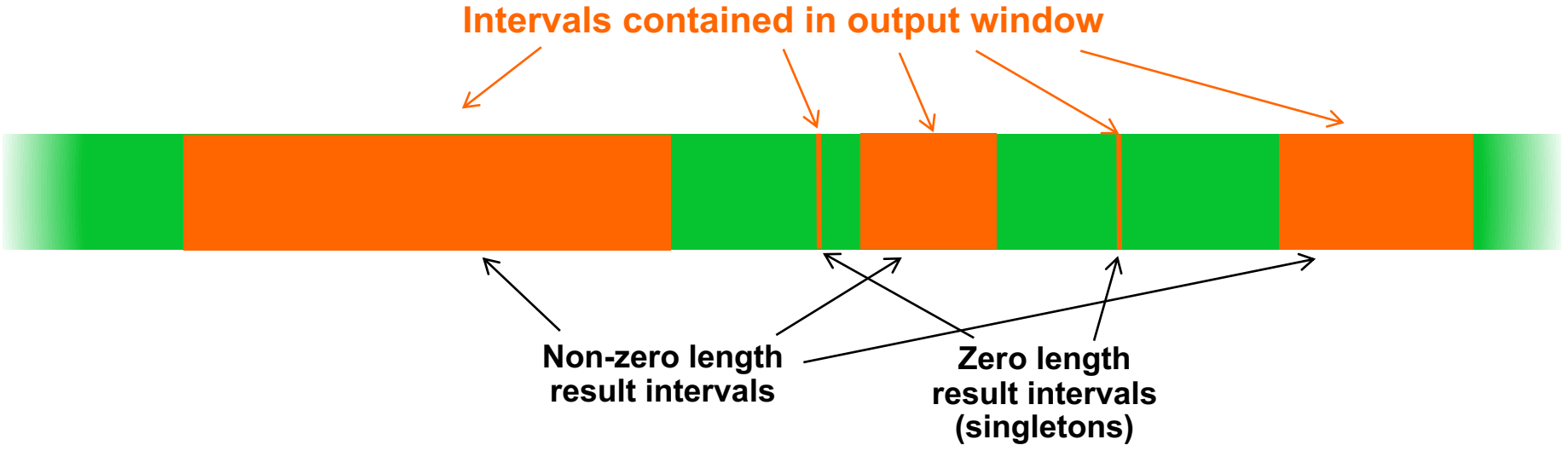
Using Time Windows in GF

- **GF uses a SPICE window to:**
  - **confine the time bounds over which your search is to take place**



Using Time Windows in GF

- **GF uses SPICE windows for input and output**
  - **Input: confine the time bounds over which your search is to take place**
  - **Output: contain the time intervals that meet the search criteria**
    - » **There may be none, one or multiple result intervals**
    - » **The result intervals can be of non-zero or zero length**
      - **A zero-length interval is simply an epoch–an instant in time**



Cascading Search Using Multiple SPICE Windows

- **The result window (the output) from one search can be used as the confinement window (the input) for a subsequent search.**
  - **This is often a convenient and efficient way of performing searches for times when multiple constraints are met.**
  - **This technique can be used to accelerate searches in cases where an initial, fast search can be performed to produce a small confinement window for a second, slower search.**
    - » **See the next chart and the example program "CASCADE" in the Geometry Finder Required Reading document**

Cascading Search Example

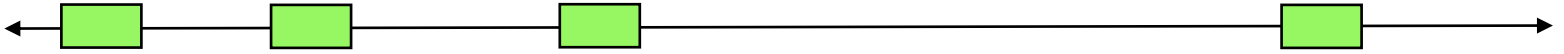
Example: accelerate a solar occultation search.

First search for times when the angular separation of the Sun and Moon, as seen from an earth station, is less than 3 degrees.

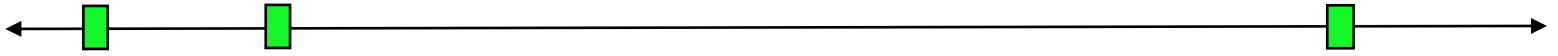
Use the result window of the angular separation search as the confinement window of an occultation search.

Because the angular separation search is much faster than would be the occultation search on the original confinement window, the total search time is greatly reduced.

Original confinement window (“A”)



Result of angular separation search: second confinement window (“B”)



Window “C”: result of occultation search performed on window “B”

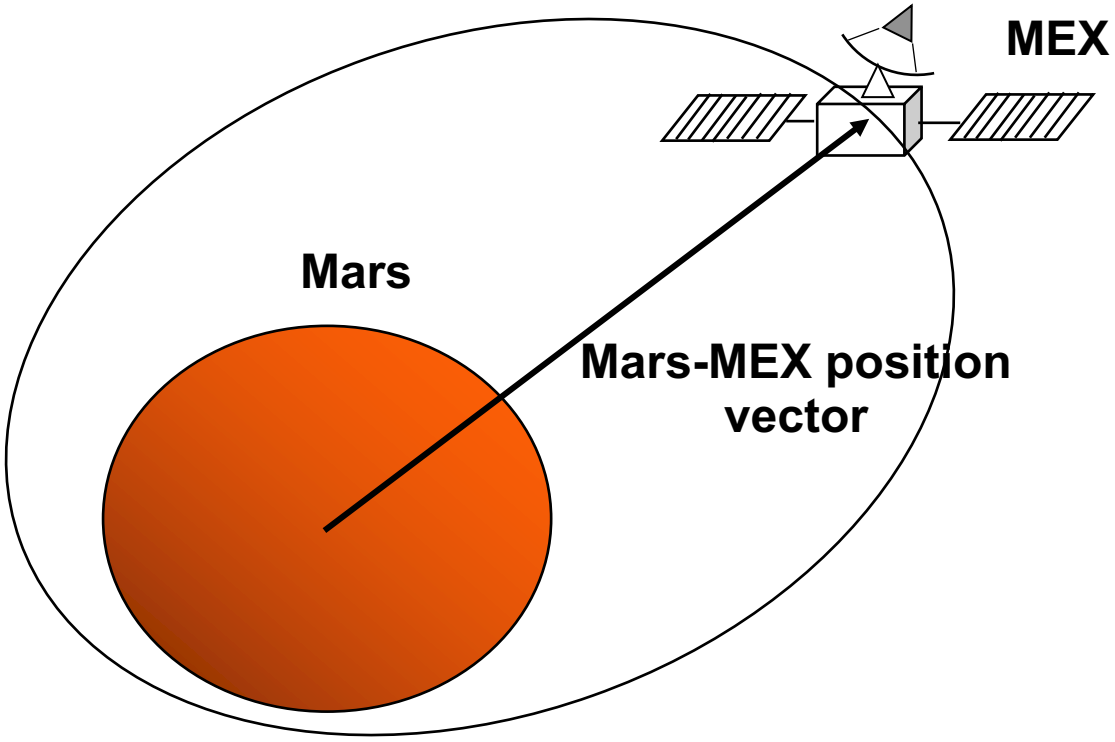
GF Documentation

- The GF module headers contain complete example programs for each GF API routine
- The GF Required Reading document (gf.req) contains lots of details
- Documentation on SPICE windows:
  - The WINDOWS Required Reading windows.req
  - The Other Functions tutorial
  - API documentation for SPICE window routines

GF Search Examples

Distance is Local Maximum (or Minimum)

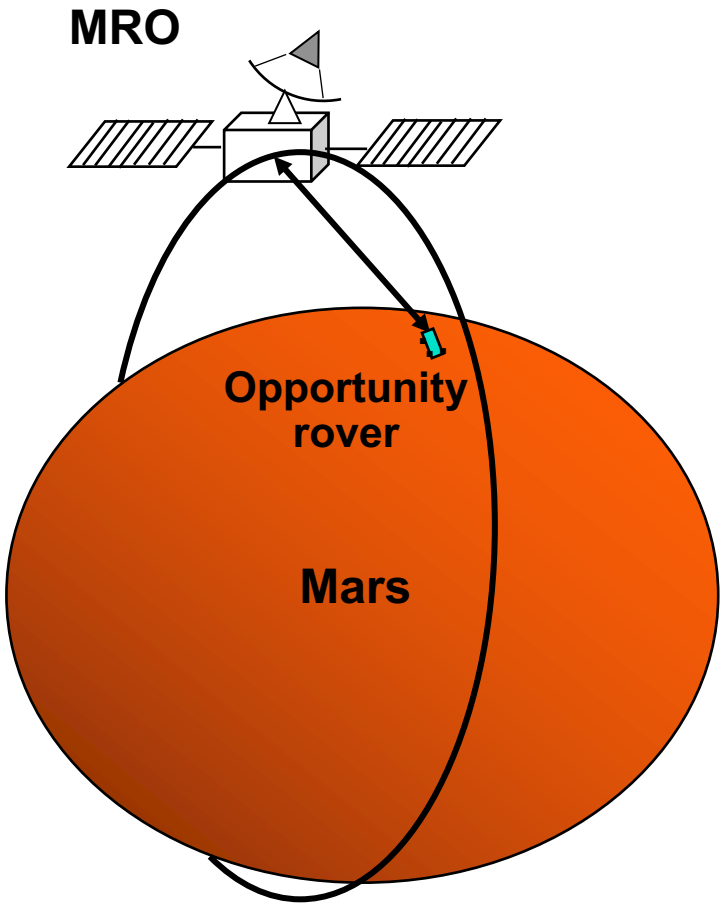
Find the times of apoapse of the Mars Express Orbiter (MEX)



API: GFDIST

Distance Within a Range

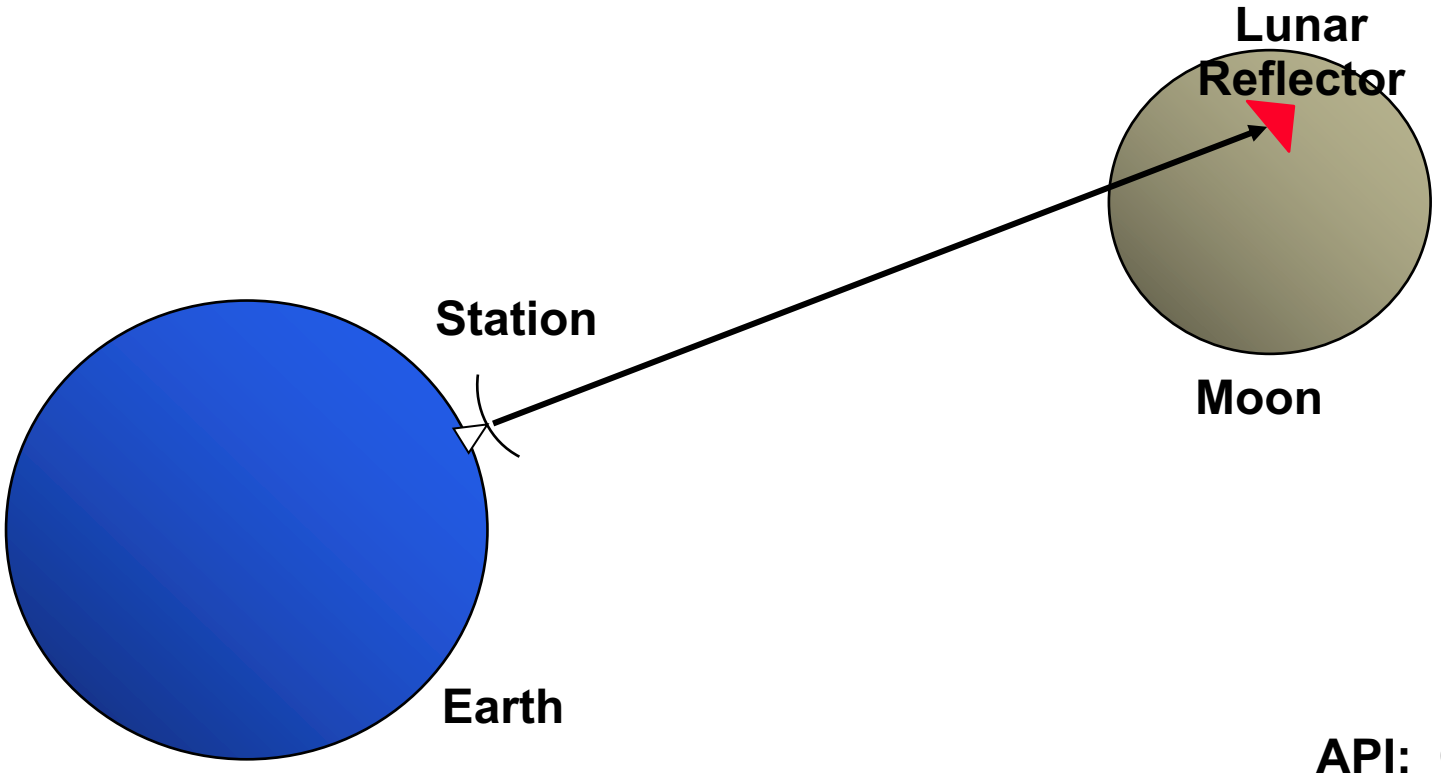
Find the time periods when the Mars Reconnaissance Orbiter (MRO) is within 500km of the Opportunity rover.



API: GFDIST

Range Rate Extremum

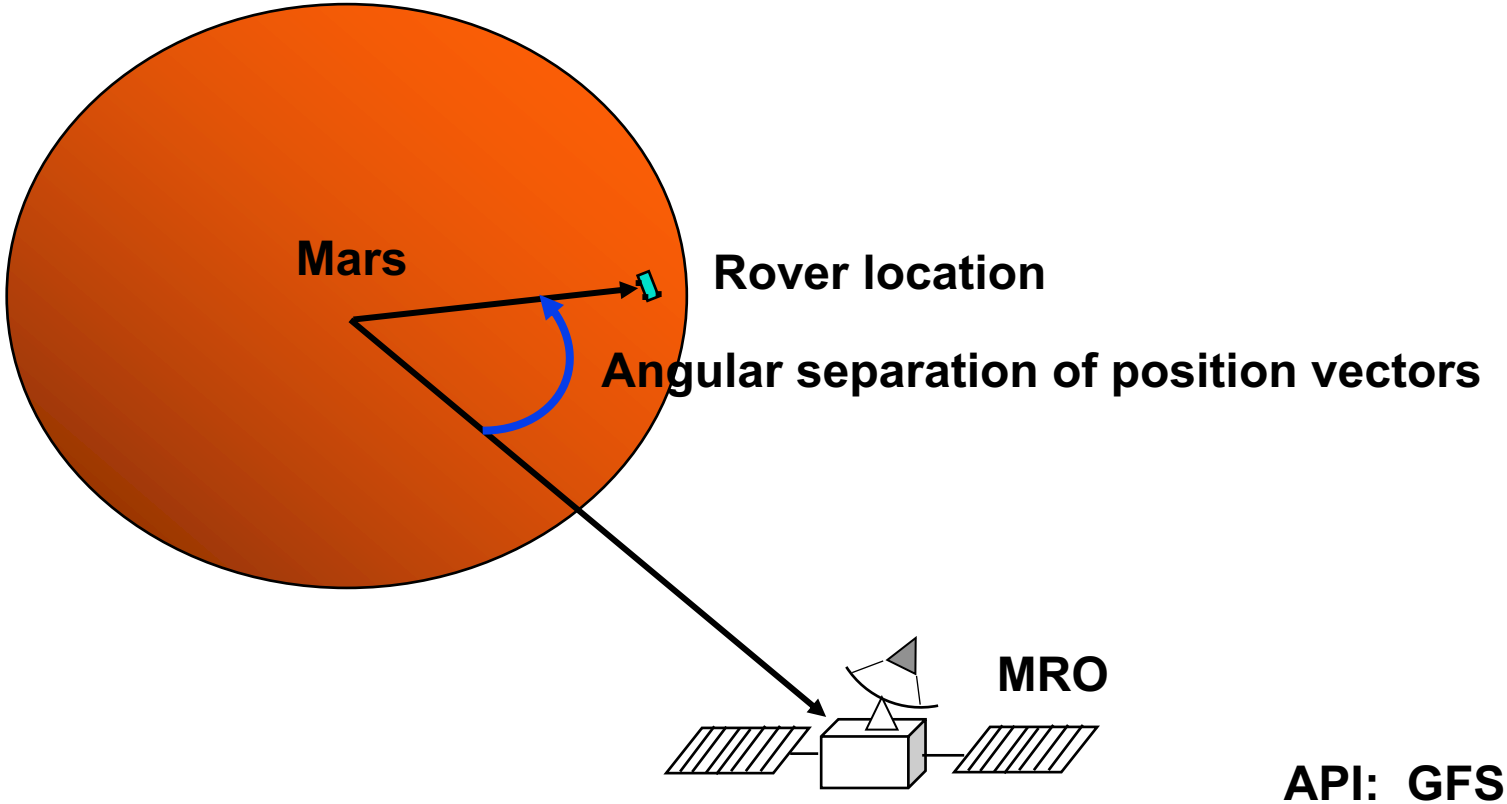
Find the time periods when the range rate of a lunar reflector, as seen by an Earth station, attains an absolute extremum.



API: GFRR

Angular Separation Inequality Search -1

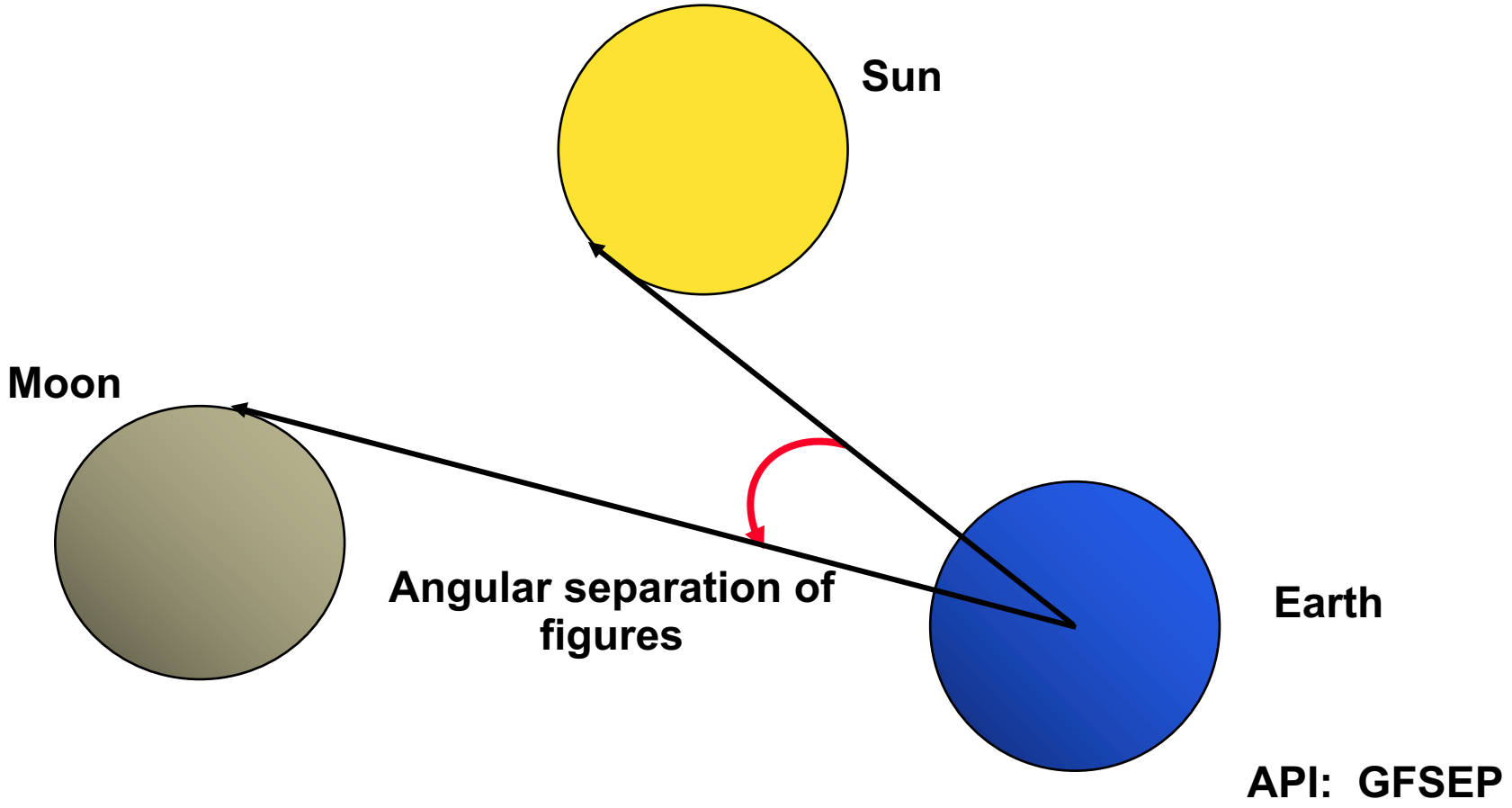
Find the time periods when the angular separation of the Mars-to Mars Reconnaissance Orbiter (MRO) and Mars-to-Opportunity Rover position vectors is less than 3 degrees. Both targets are modeled as points.



API: GFSEP

Angular Separation Inequality Search -2

Find the time periods when the angular separation of the figures of the Moon and Sun, as seen from the Earth, is less than 1 degree. Both targets are modeled as spheres.

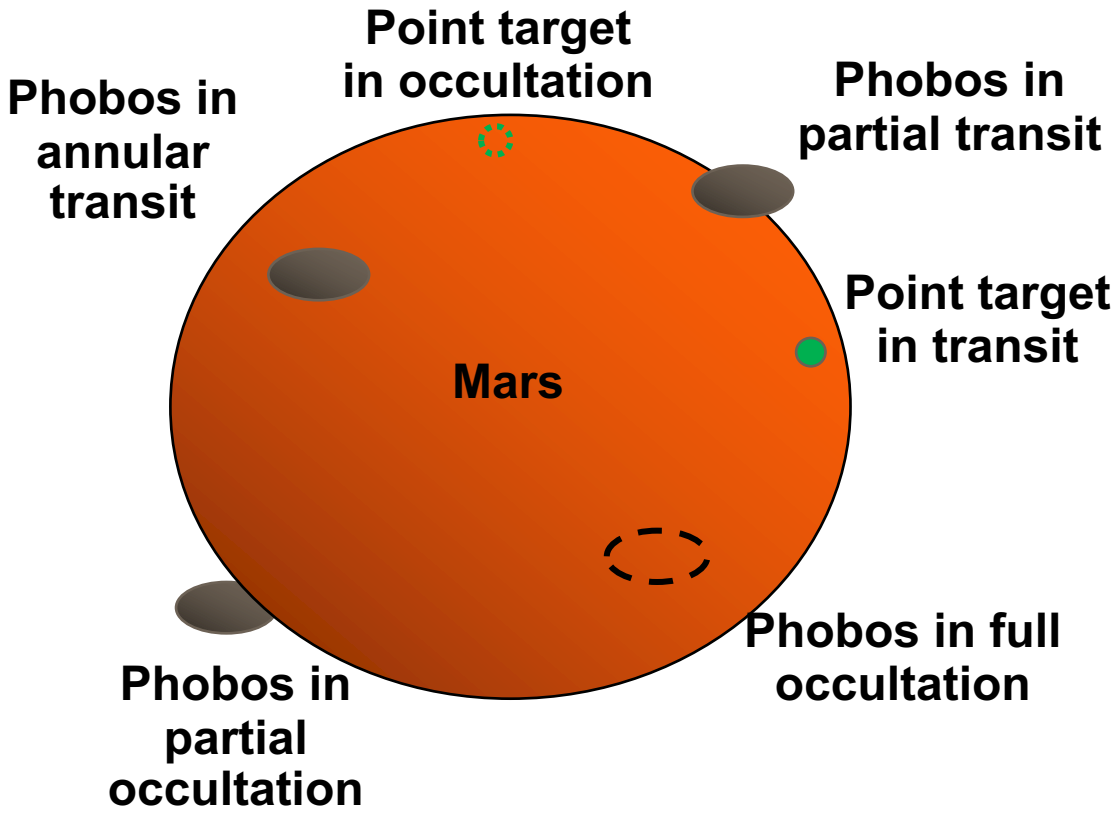


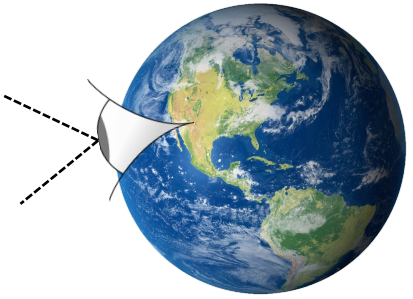
API: GFSEP

Occultation/Transit Search

Navigation and Ancillary Information Facility

Find the ingress and egress times of an occultation of Phobos by Mars, as seen from Earth. Phobos and Mars are modeled as triaxial ellipsoids; a spacecraft is modeled as a point target.



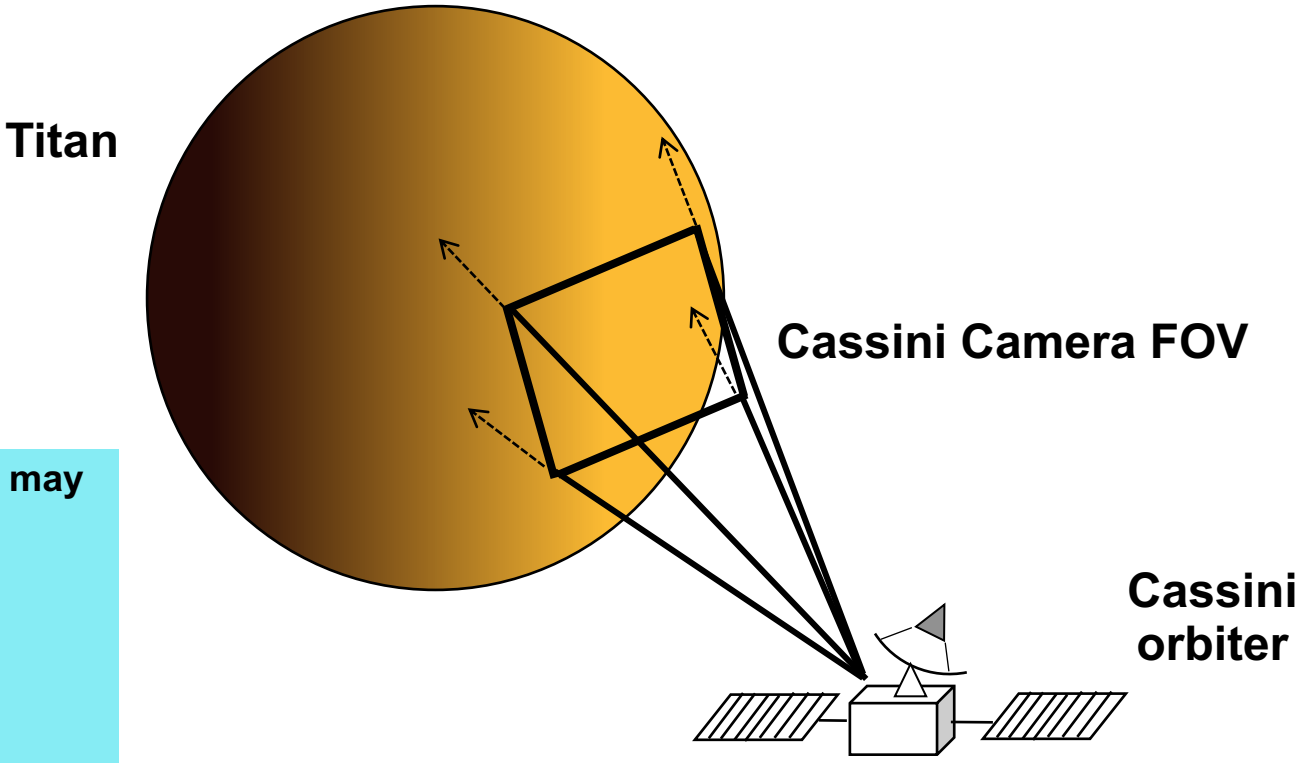


Earth

API: GFOCLT

Target in Field of View

Find the time periods when Titan appears in the FOV of the Cassini ISS Narrow Angle Camera (NAC). The target shape is modeled as an ellipsoid. (Point targets are also supported.)



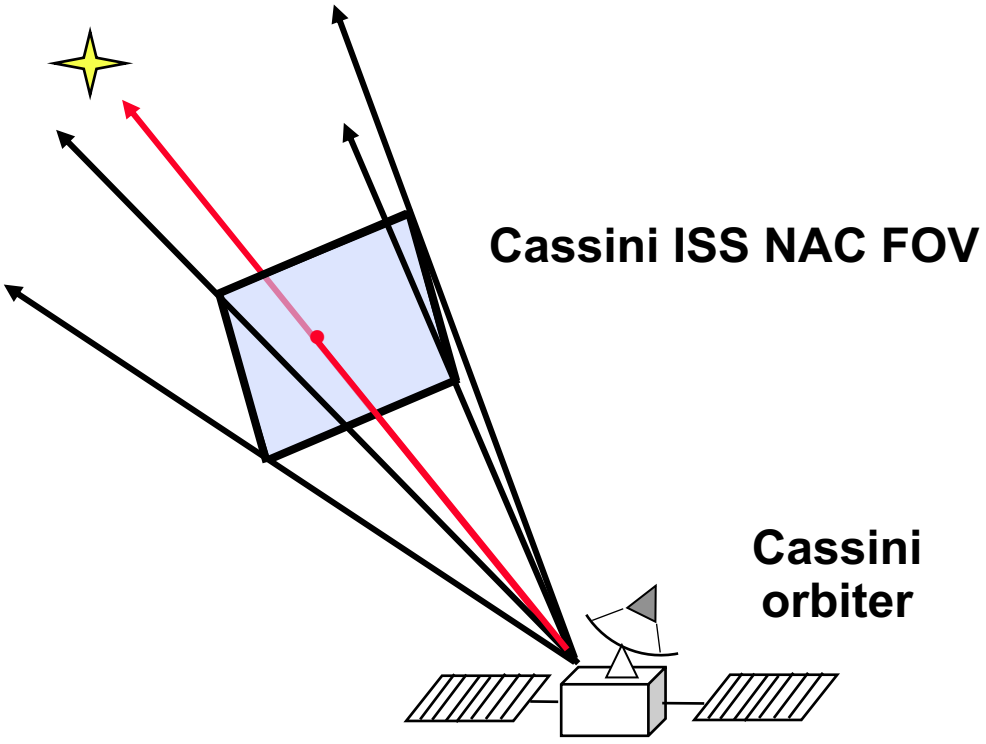
The FOV shape may be any of:

- Rectangle
- Circle
- Ellipse
- Polygon

API: GFTFOV

Ray in FOV Search

Find the time periods when a star appears in the FOV of the Cassini ISS Narrow Angle Camera (NAC). The target direction is modeled as a ray, optionally corrected for stellar aberration.



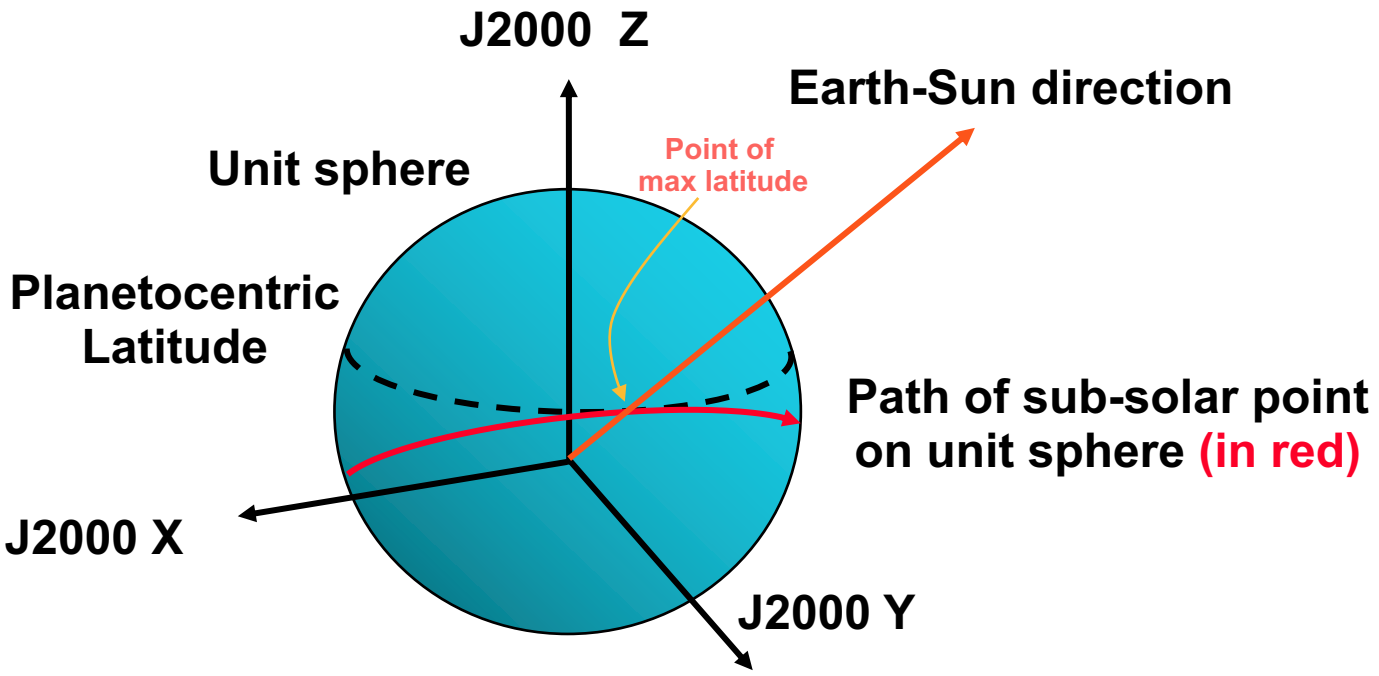
The FOV shape may be any of:

- Rectangle
- Circle
- Ellipse
- Polygon

API: GFRFOV

Position Coordinate Local Extremum Search

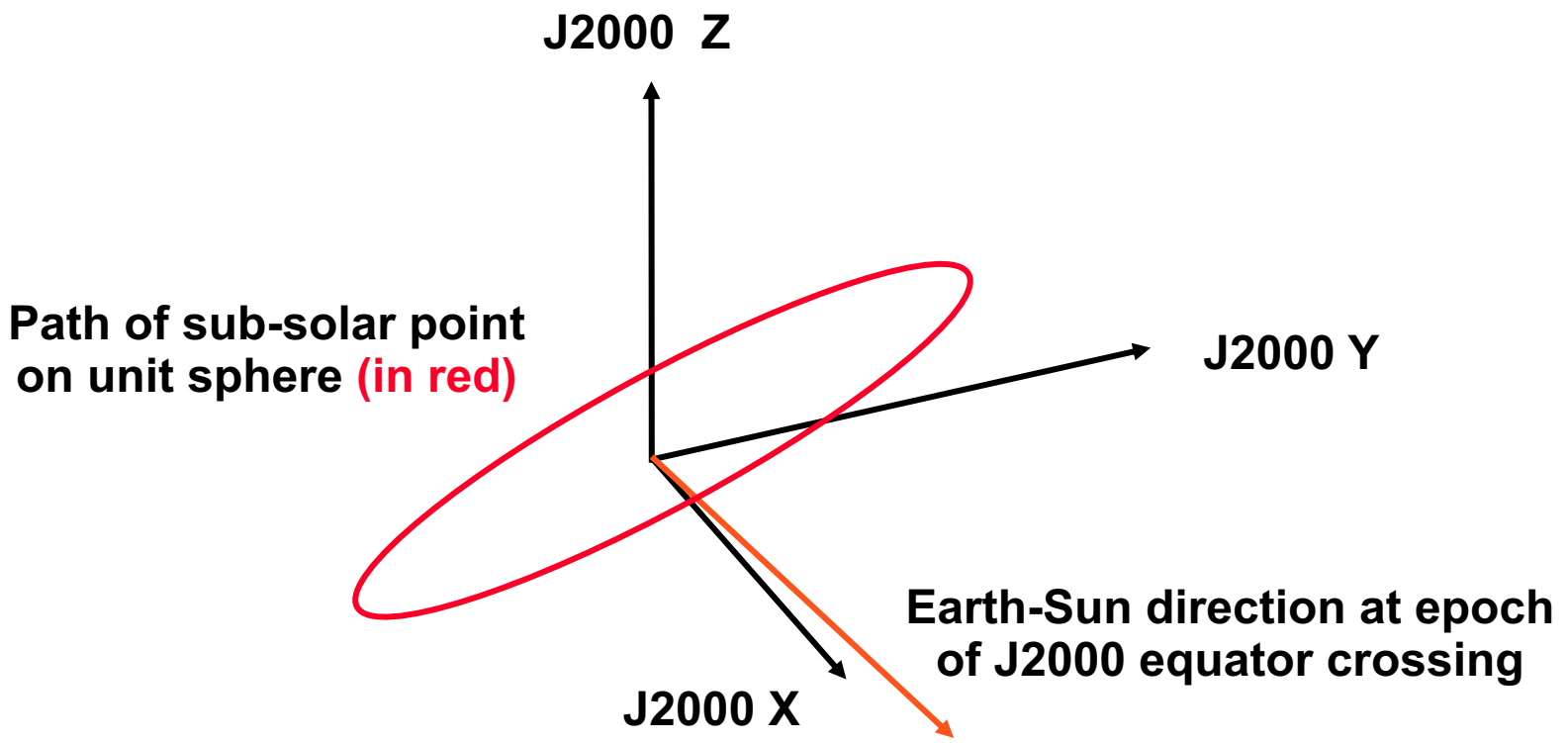
Find the time(s) at which the planetocentric latitude of the Earth-Sun vector, expressed in the J2000 frame, is at a local maximum.



API: GFPOSC

Position Coordinate Equality Search

Find the time(s) at which the Z component of the Earth-Sun vector, expressed in the J2000 frame, is 0.

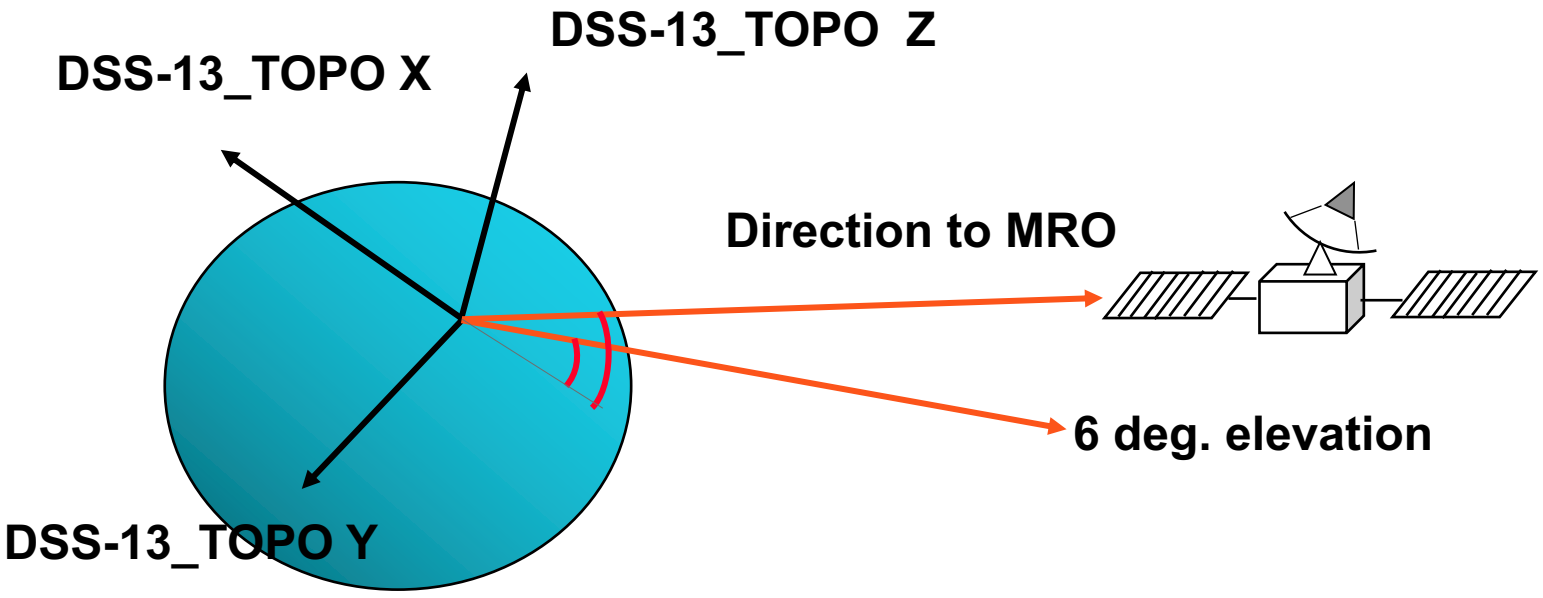


API: GFPOSC

Position Coordinate Inequality Search -1

Navigation and Ancillary Information Facility

Find the time periods when the elevation of the DSS-13 to Mars Reconnaissance Orbiter (MRO) spacecraft vector, expressed in the DSS-13 topocentric frame, is greater than 6 degrees.

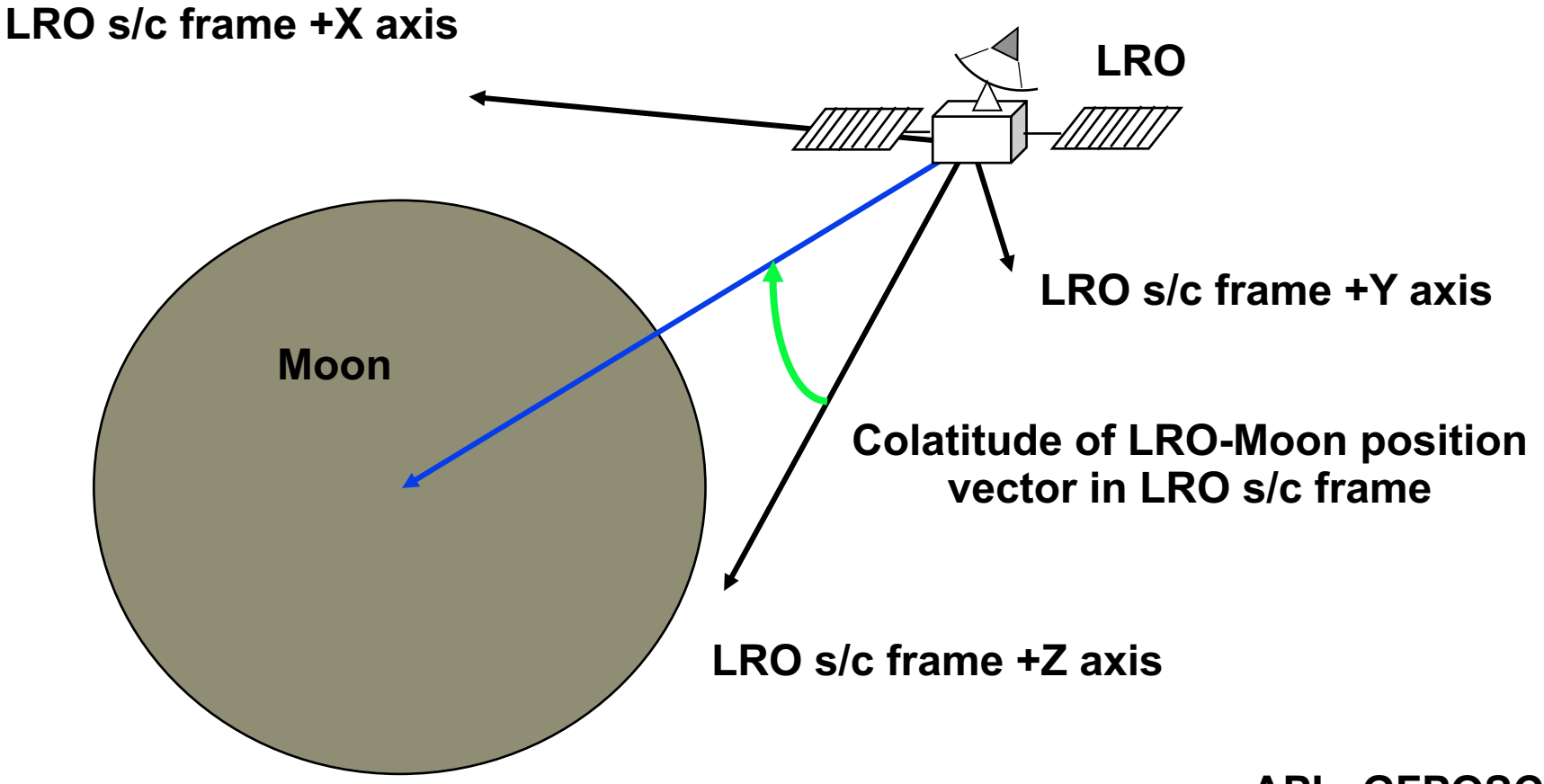


API: GFPOSC

Position Coordinate Inequality Search -2

Navigation and Ancillary Information Facility

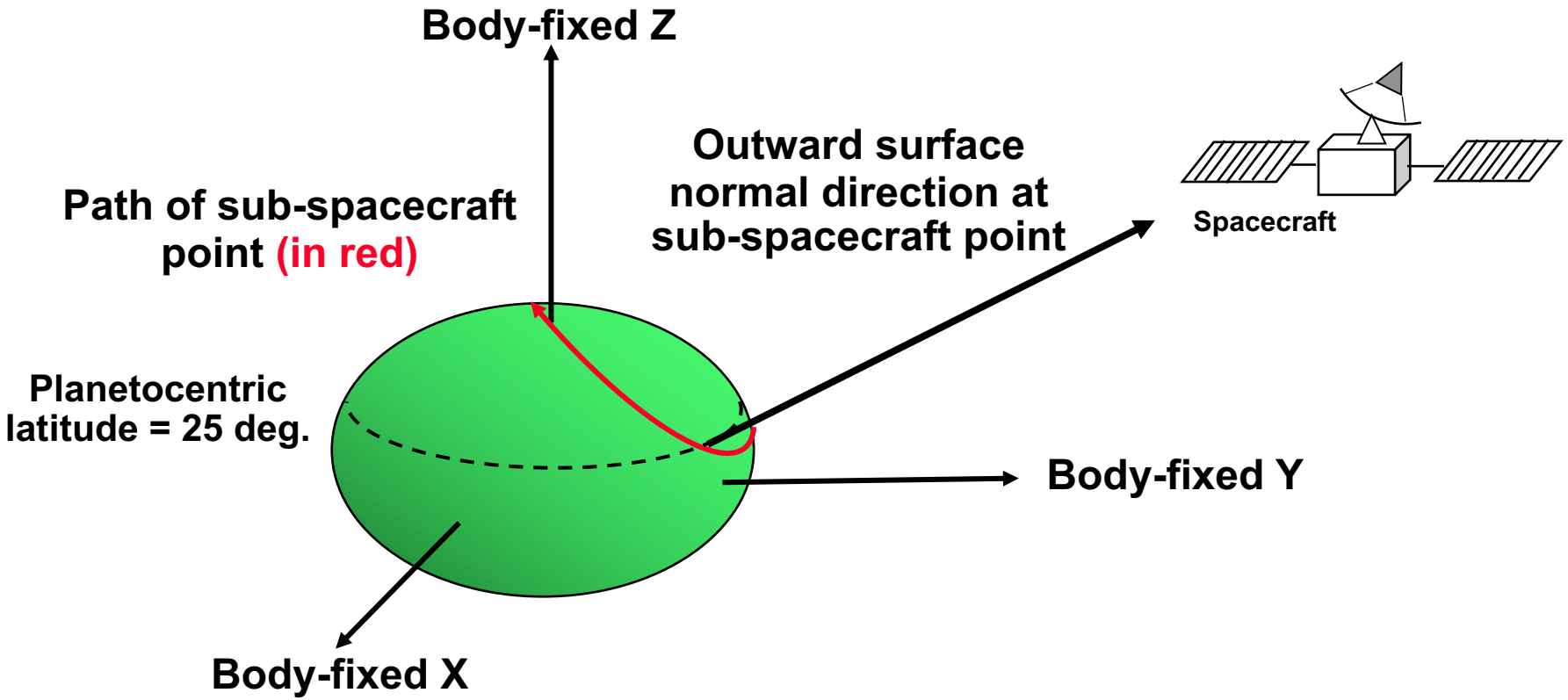
Find the time periods when the Lunar Reconnaissance Orbiter's (LRO) off-nadir angle of the +Z axis exceeds 5 degrees.



API: GFPOSC

Sub-Observer Point Coordinate Equality Search

Find the time(s) at which the planetocentric latitude of the sub-spacecraft point, using the "near point" definition, is 25 degrees.

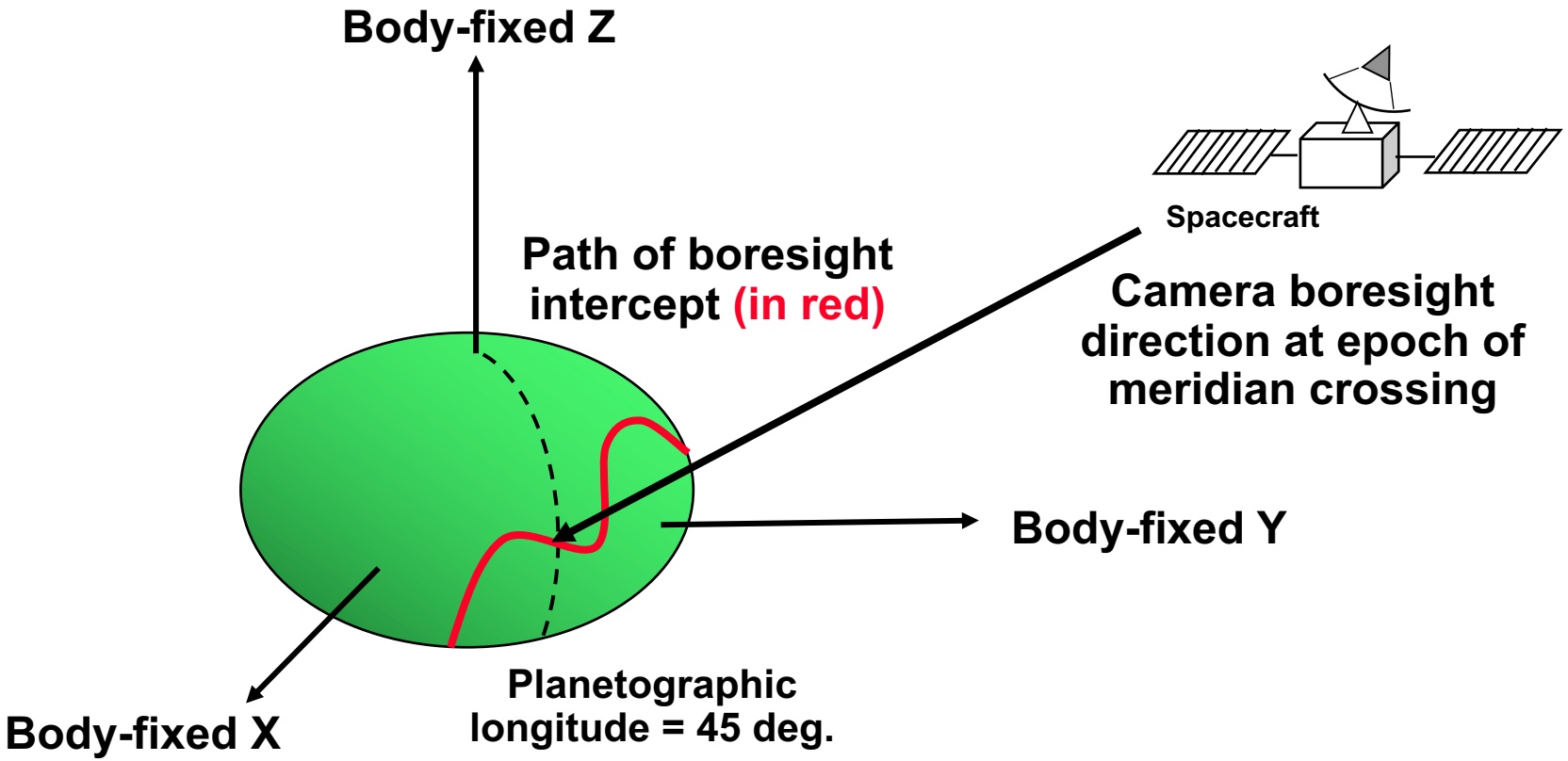


API: GFSUBC

Surface Intercept Coordinate Equality Search

Navigation and Ancillary Information Facility

Find the time(s) at which the planetographic longitude of a given camera boresight surface intercept is 45 degrees.

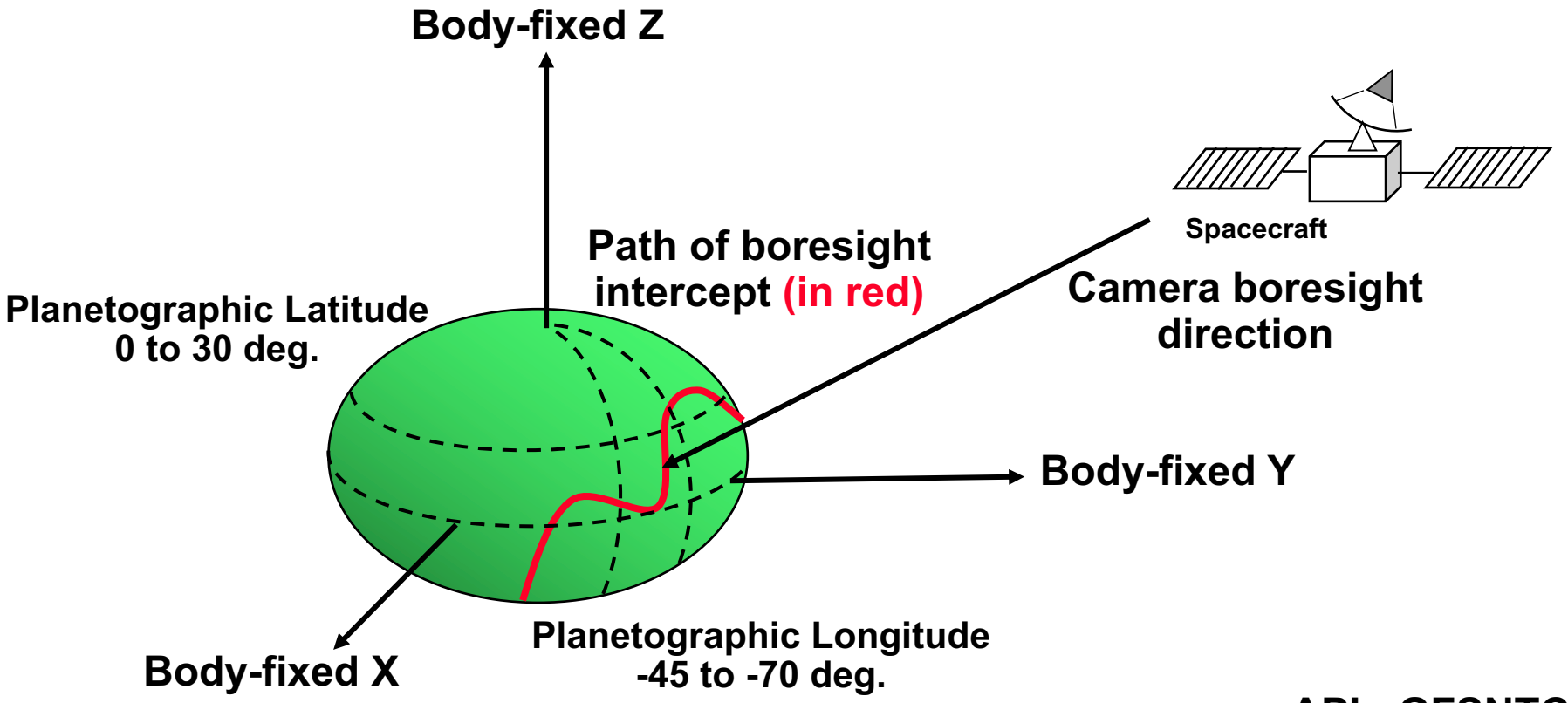


API: GFSNTC

Surface Intercept "Box" Search

Find the time periods when the planetographic longitude of a camera boresight surface intercept is between -70 and -45 degrees, and the intercept latitude is between 0 and 30 degrees.

The solution requires four (cascading) inequality searches.



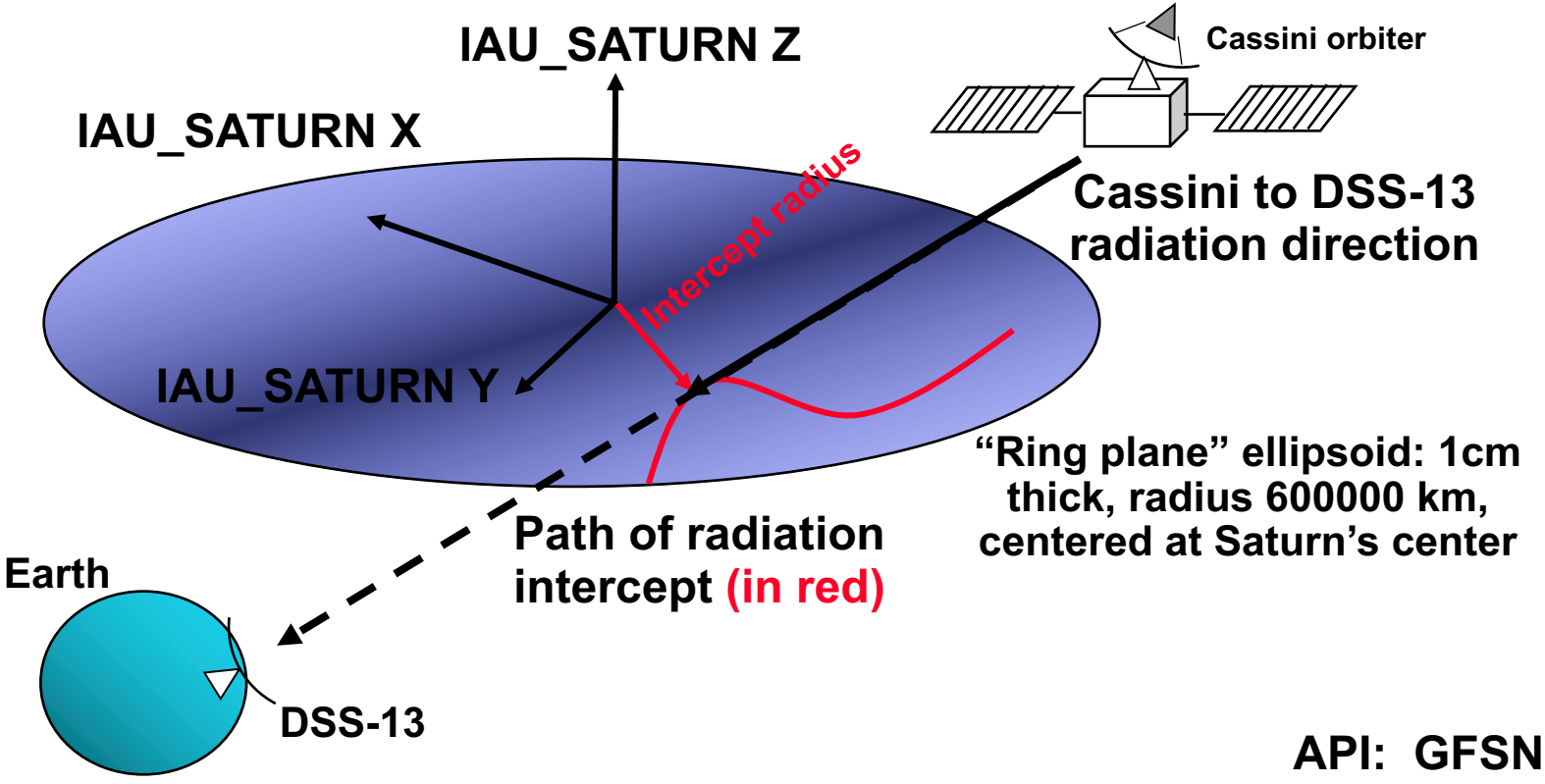
API: GFSNTC

Surface Intercept Coordinate Equality Search

Navigation and Ancillary Information Facility

Find the time at which the ring plane intercept of the Cassini to DSS-13 vector, corrected for transmission light time (stellar aberration correction is unnecessary), has radius 300000km.

The solution requires a dynamic frame for which one axis points along the radiation path.

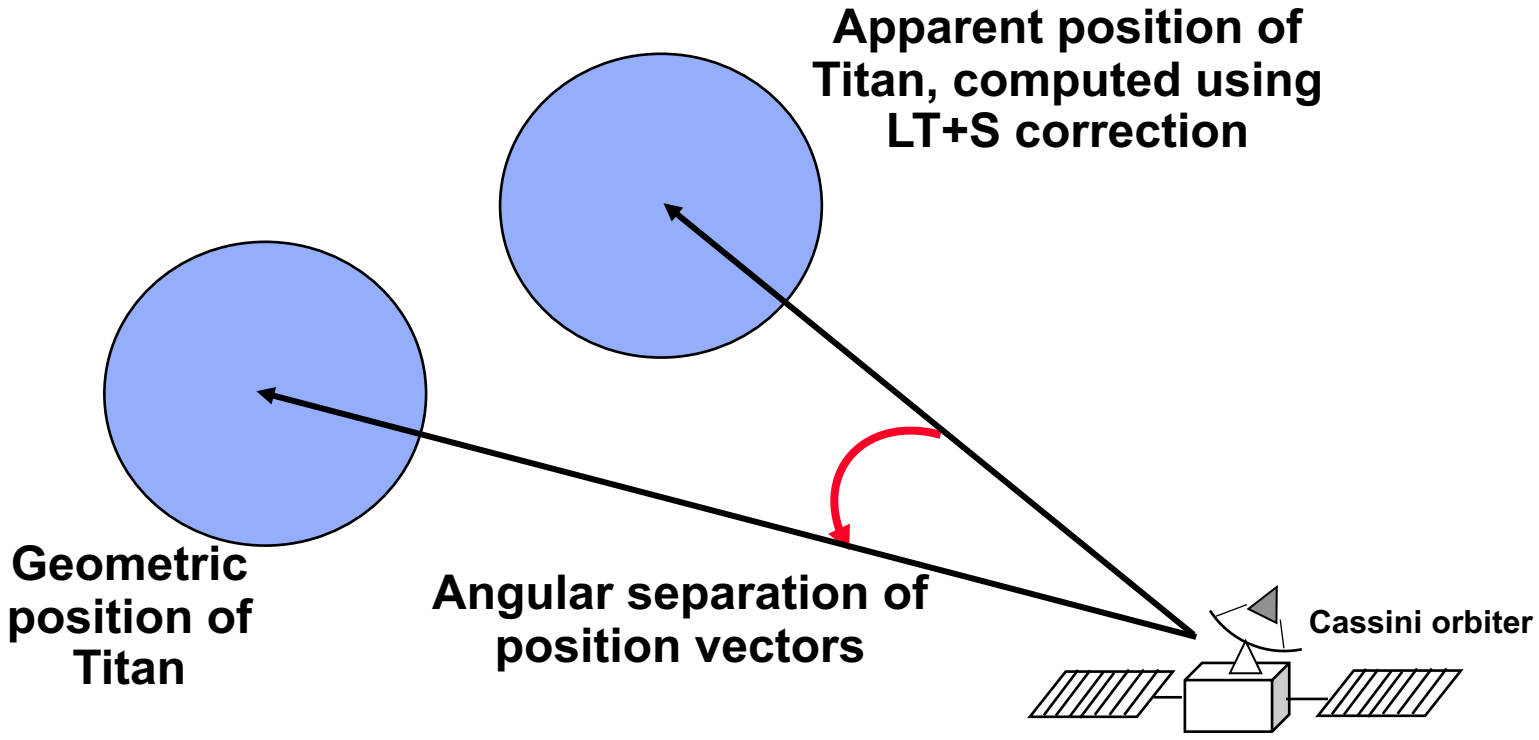


API: GFSNTC

User-Defined Quantity Extremum Search

Navigation and Ancillary Information Facility

Find the time periods when the angular separation of the geometric and apparent positions of Titan as seen from the Cassini orbiter attains an absolute maximum.



API: GFUDS

Geometric Search Types and Constraints

Types of Geometric Quantities

- The GF subsystem deals with two types of geometric quantities:
  - "Binary state functions": functions of time that can be "true" or "false." Examples:
    - » Occultation state, such as: "Titan is fully occulted by Saturn at time t"
    - » Visibility state: A target body or object modeled as a ray (for example, a star) is visible in a specified instrument FOV at time t
  - Scalar numeric functions of time, for example
    - » Observer-target distance
    - » Latitude or longitude of an observer-target vector, sub-spacecraft point, or ray-surface intercept point
    - » Angular separation of two target bodies as seen by an observer

Binary State Searches

- **Binary state searches find times when a specified binary state function takes the value "true."**
  - **SPICE window arithmetic can be used to find the times when a binary state function is "false."**

Numeric Searches

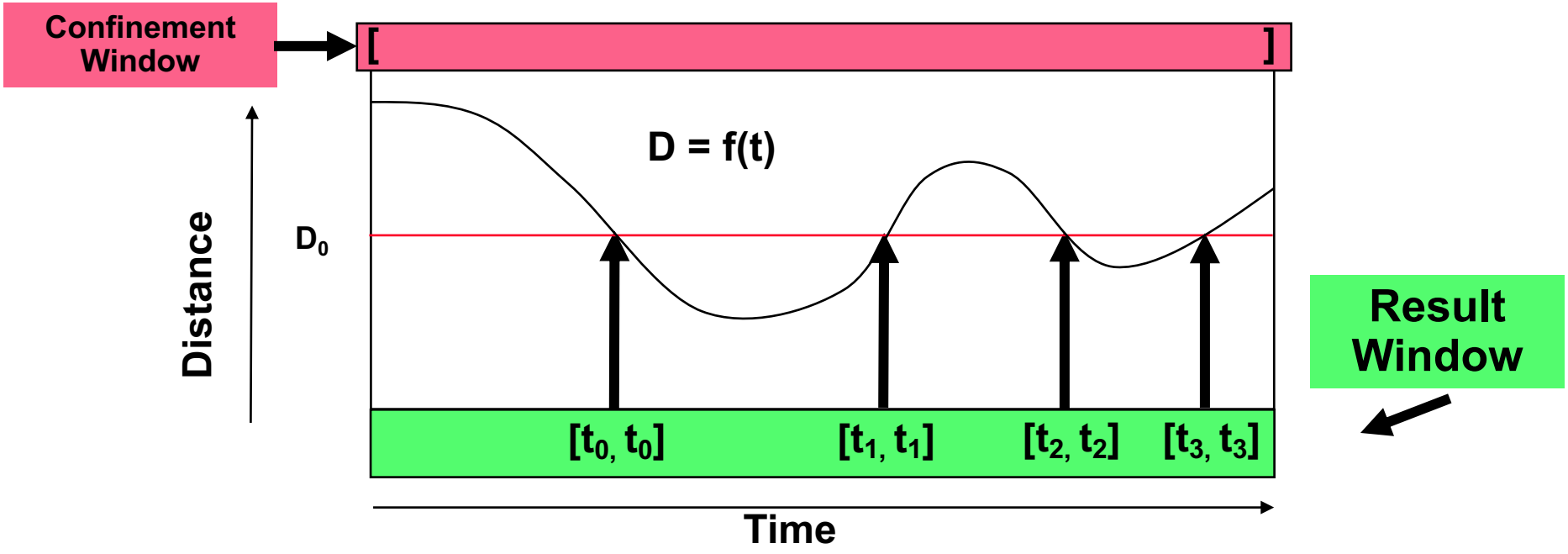
- **Numeric searches find times when a scalar numeric quantity satisfies a mathematical constraint. The supported constraints are:**
  - **The function**
    - » **equals a specified reference value.**
    - » **is less than a specified reference value.**
    - » **is greater than a specified reference value.**
    - » **is at a local maximum.**
    - » **is at a local minimum.**
    - » **is at an absolute maximum.**
    - » **is at an absolute minimum.**
    - » **is at an "adjusted" absolute maximum: the function is within a given tolerance of the absolute maximum.**
    - » **is at an "adjusted" absolute minimum: the function is within a given tolerance of the absolute minimum.**
- **Examples for a Distance search follow.**

Solve Distance Equality

Find times at which Distance = D_0

GF API input arguments defining constraint:

```
RELATE   =   '='
ADJUST   =   0.D0
REFVAL   =   D0
```

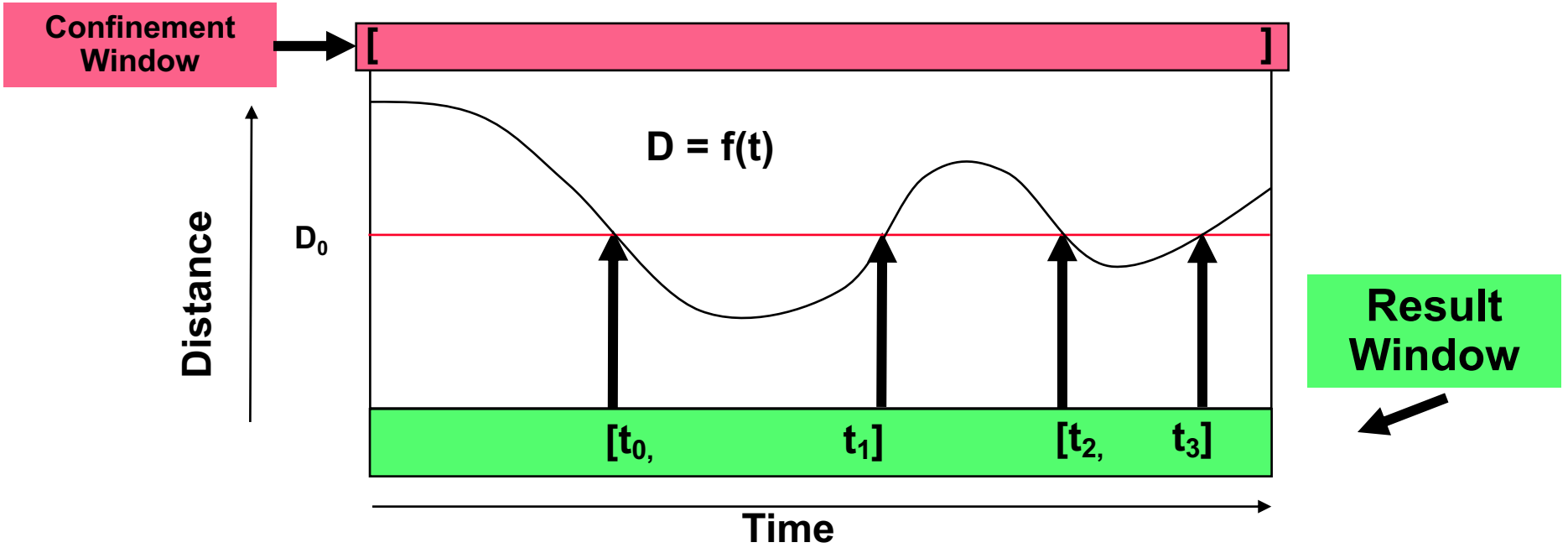


Solve Distance < Inequality

Find times during which Distance < D_0

GF API input arguments defining constraint:

```
RELATE   =   '<'
ADJUST   =   0.D0
REFVAL   =   D0
```

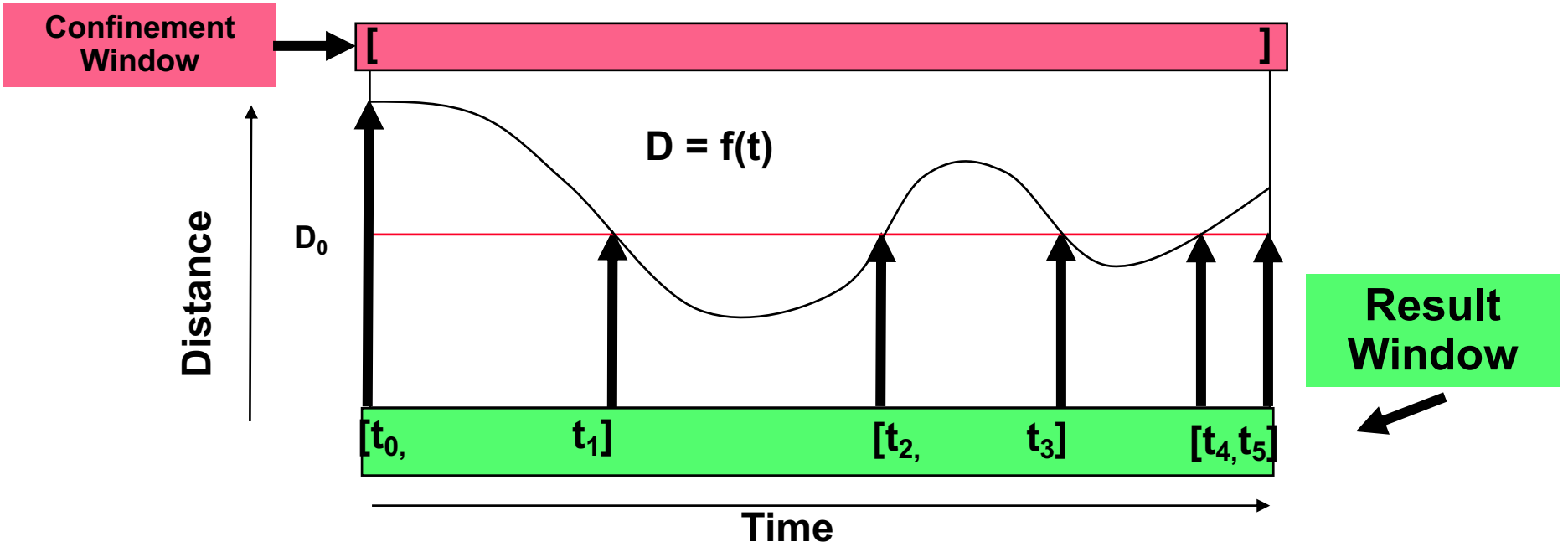


Solve Distance > Inequality

Find times during which Distance > D_0

GF API input arguments defining constraint:

```
RELATE  =  '>'
ADJUST  =  0.D0
REFVAL  =  D0
```



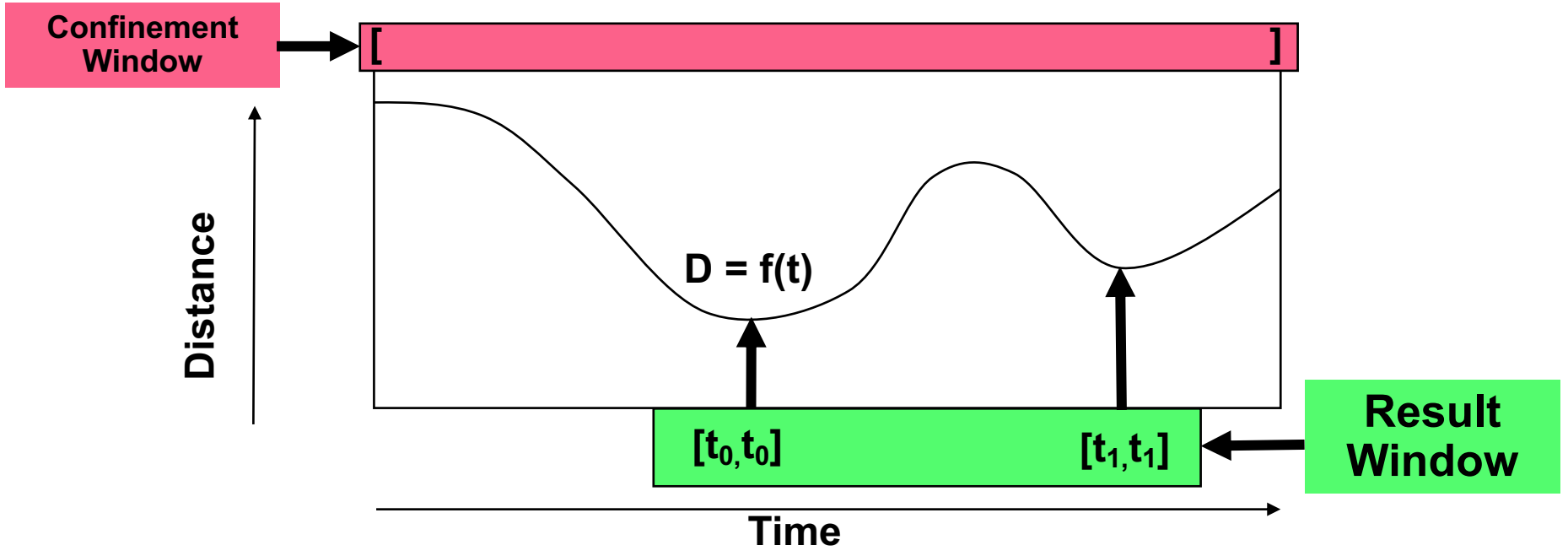
Find Local Minima

Find times at which Distance is a local minimum.

GF API input arguments defining constraint:

```
RELATE = 'LOCMIN'
ADJUST = 0.D0
REFVAL = 0.D0
```

(REFVAL is not used in this case
but should be initialized for portability)



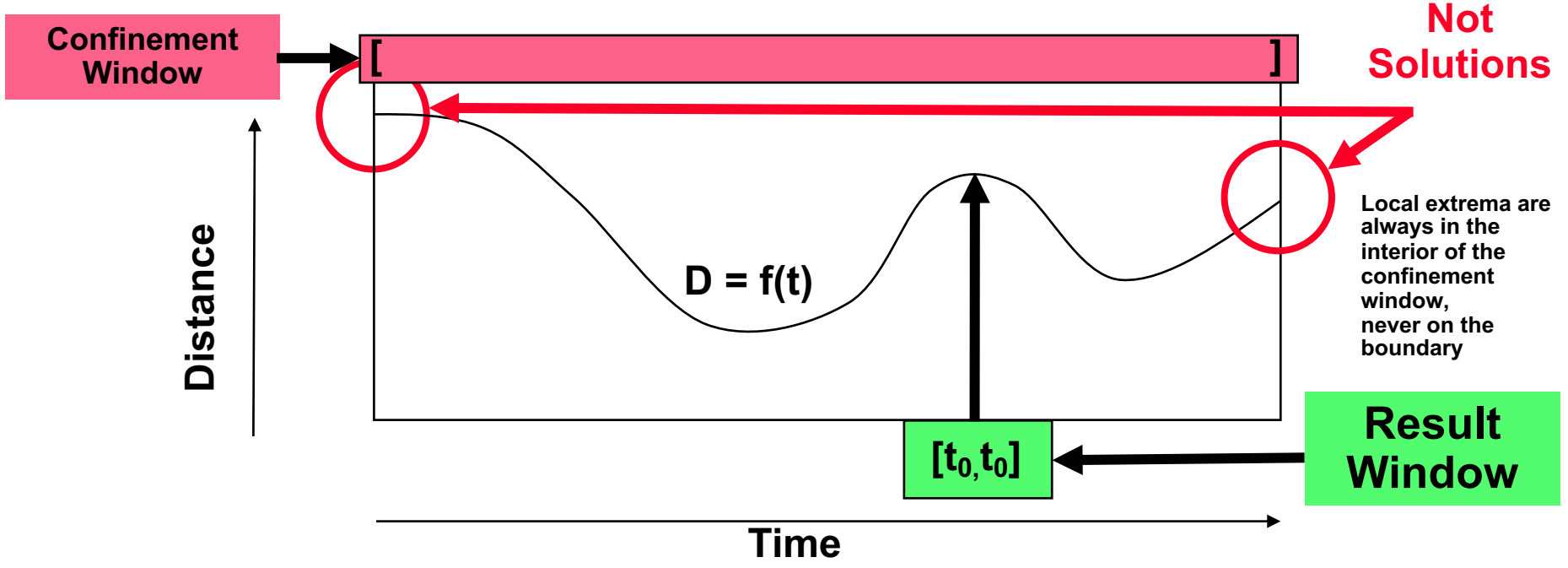
Find Local Maxima

Find times at which Distance is a local maximum.

GF API input arguments defining constraint:

```
RELATE = 'LOCMAX'
ADJUST = 0.D0
REFVAL = 0.D0
```

(REFVAL is not used in this case
but should be initialized for portability)

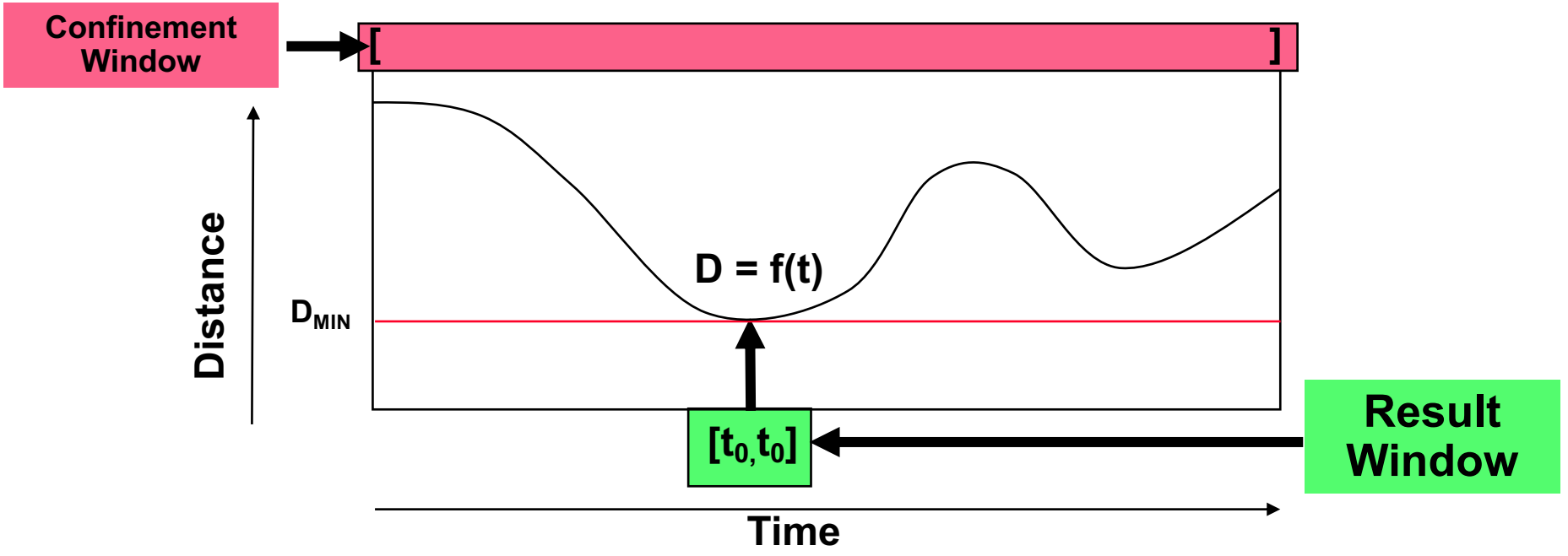


Find Absolute Minimum

Find the time at which Distance is an absolute minimum.

GF API input arguments defining constraint:

```
RELATE = 'ABSMIN'
ADJUST = 0.D0          (REFVAL is not used in this case
REFVAL = 0.D0          but should be initialized for portability)
```



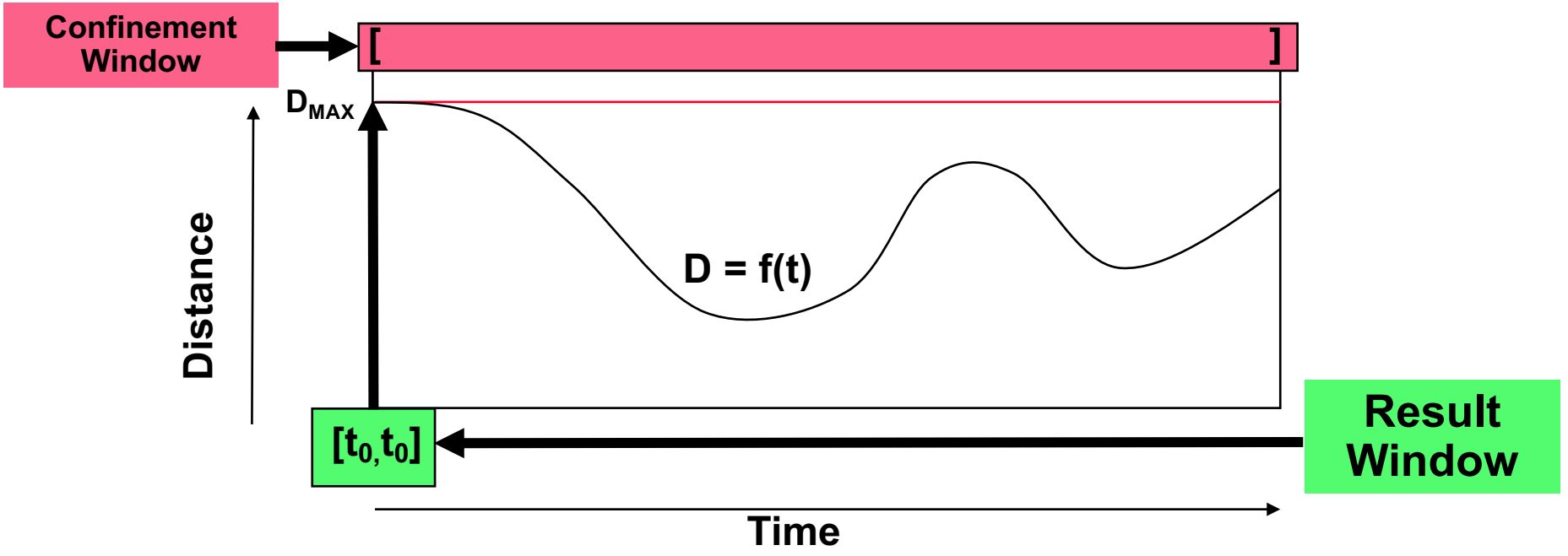
Find Absolute Maximum

Find the time at which Distance is an absolute maximum.

GF API input arguments defining constraint:

```
RELATE = 'ABSMAX'
ADJUST = 0.D0
REFVAL = 0.D0
```

(REFVAL is not used in this case but should be initialized for portability)



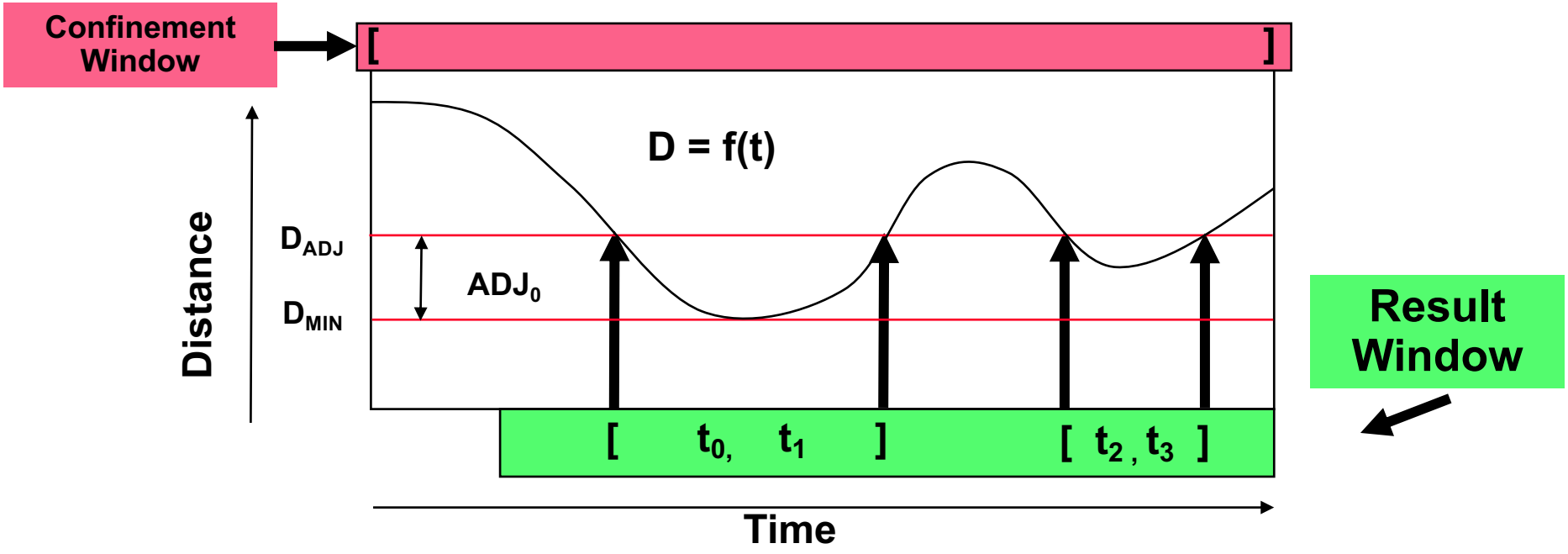
Find Adjusted Absolute Minimum

Navigation and Ancillary Information Facility

Find times at which Distance < $D_{ADJ} = D_{MIN} + ADJ_0$

GF API input arguments defining constraint:

```
RELATE = 'ABSMIN'
ADJUST = ADJ0    (ADJ0 > 0)
REFVAL = 0.D0    (REFVAL is not used in this case but should be initialized for portability)
```



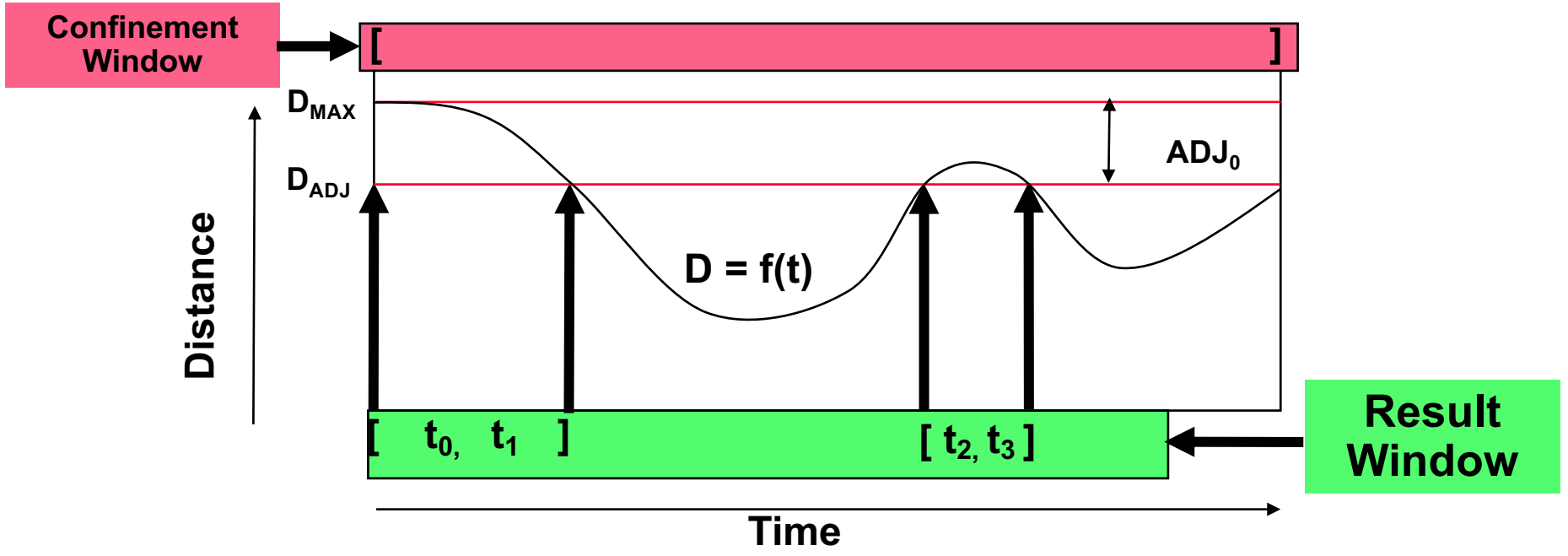
Find Adjusted Absolute Maximum

Navigation and Ancillary Information Facility

Find times at which Distance > $D_{ADJ} = D_{MAX} - ADJ_0$

GF API input arguments defining constraint:

```
RELATE = 'ABSMAX'
ADJUST = ADJ0    (ADJ0 > 0)
REFVAL = 0.D0    (REFVAL is not used in this case but should be initialized for portability)
```



More Details

- GF mid-level APIs
- Root Finding, including step size selection
- Workspace
- API Example: GFDIST

GF Mid-Level API Routines

- **The Fortran and C SPICE Toolkits provide some mid-level APIs that provide additional capabilities:**
  - **Progress reporting, which can be customized by the user**
  - **Interrupt handling which can be customized by the user**
    - » **In Fortran, no default interrupt detection is available**
  - **User-customizable search step and refinement routines**
  - **User-adjustable root finding convergence tolerance**

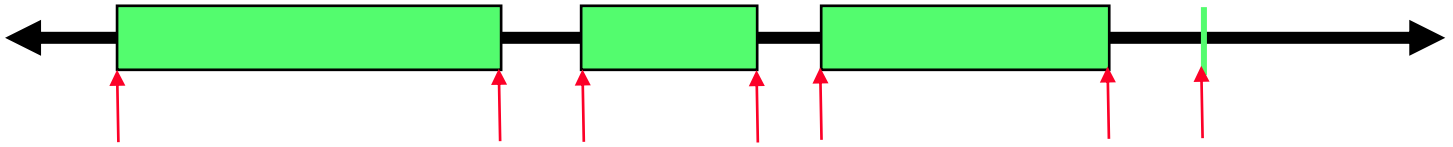
- **The GF mid-level search APIs are:**
  - **GFEVNT: All scalar numeric quantity searches**
  - **GFFOVE: Target or ray in FOV searches**
  - **GFOCCE: Occultation searches**

Root Finding

Root Finding

Navigation and Ancillary Information Facility

- To produce a final or intermediate result window, the GF subsystem must accurately locate the endpoints of the window's intervals. These endpoints are called "roots."
  - The green regions below (rectangles and vertical line segment) represent intervals of a window.
  - Roots are indicated by the red, vertical arrows.



- Elsewhere, "root finding" often refers to solving f(x) = 0.
- In the GF setting, roots are boundaries of time intervals over which a specified constraint is met.
  - Roots can be times when a binary state function changes values.
- Most popular root finding methods, e.g. Newton, secant, bisection, require the user to first "bracket" a root: that is, determine two abscissa values such that a single root is located between those values.
- The GF subsystem solves a more difficult problem: it performs a global search for roots. That is, given correct inputs, it finds all roots within a user-specified confinement window.
  - The user is not asked to bracket the roots.

Step Size Selection

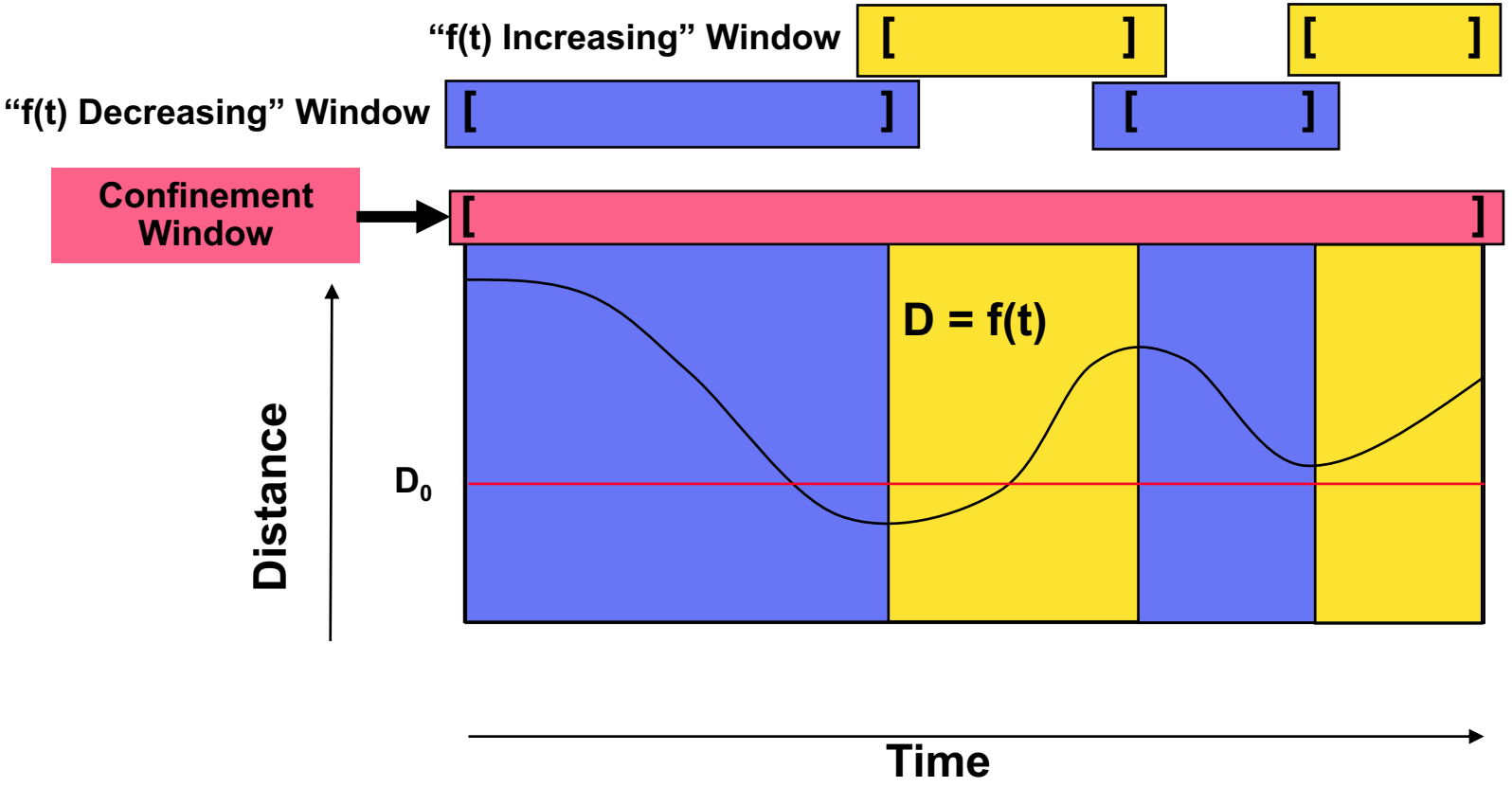
- The GF subsystem asks the user to specify a time step (often called the“step size”) that will be used to bracket roots.
  - For binary state searches, the step size is used to bracket the endpoints of the intervals of the result window.
    - » The step size must be shorter than any event of interest, and shorter than any interval between events that are to be distinguished.
  - For numeric searches, the step size is used to bracket the endpoints of the intervals of the window on which the geometric quantity is monotonically decreasing.
    - » The step size must be shorter than any interval on which the function is monotonically increasing or decreasing.
  - In both cases, the step size must be large enough so the search completes in a reasonable amount of time.

Monotone Windows -1

Example: monotone windows for a distance function

Note that:

- Extrema occur only at window interval boundaries.
- Each window interval contains at most one root of an equality condition.
- Within each window interval, the solution of an inequality is a single (possibly empty) interval.

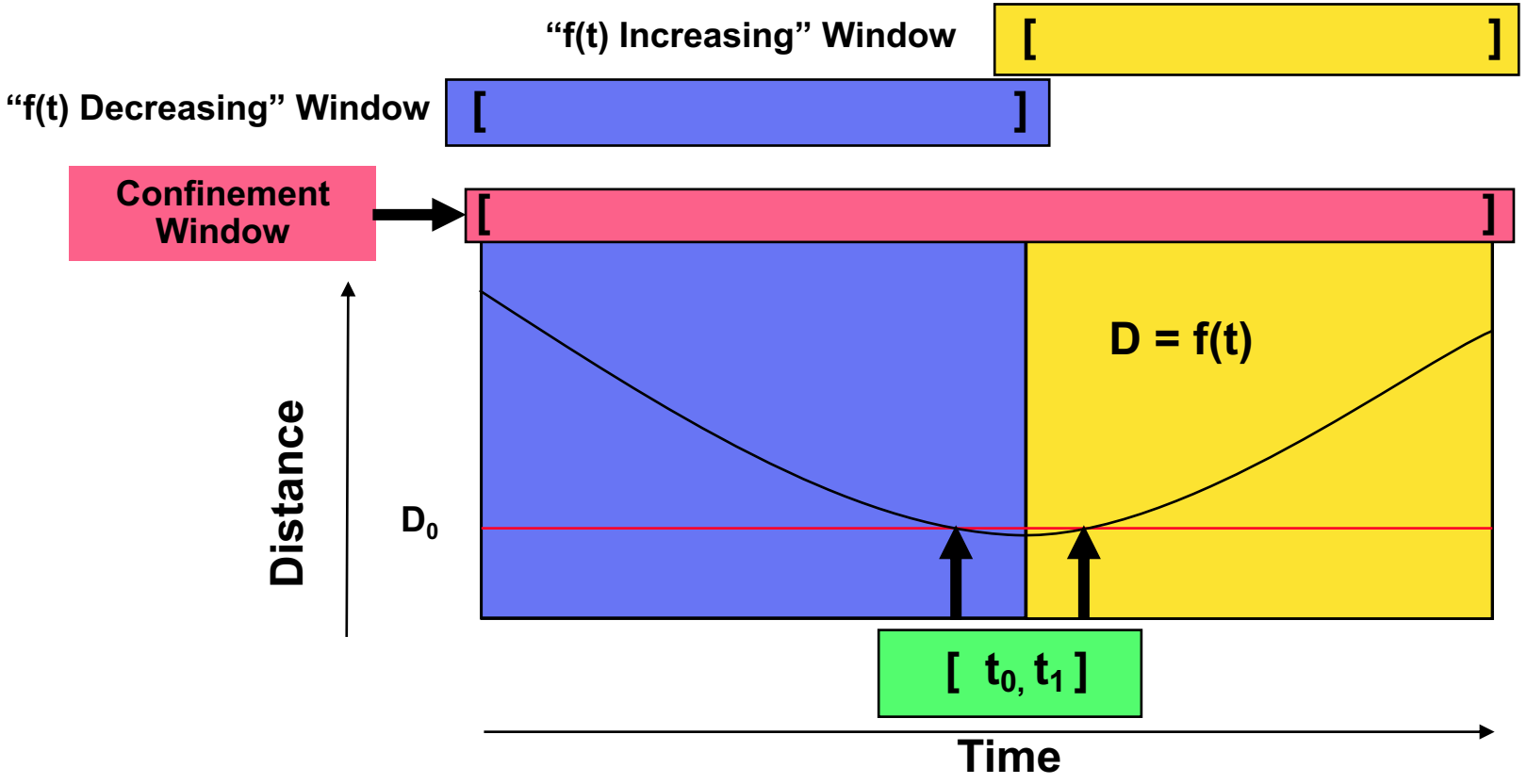


Monotone Windows-2

Navigation and Ancillary Information Facility

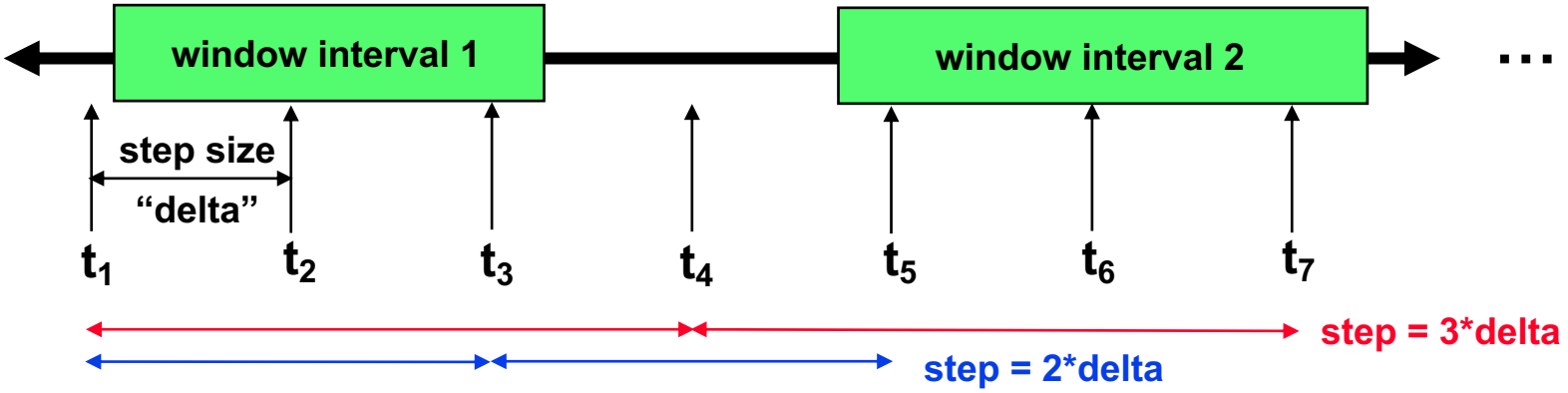
The shortest interval on which the function is monotonically increasing or decreasing may be LONGER than the event of interest.

For example, consider the search for times when $D < D_0$. The result window consists of the interval $[t_0, t_1]$ shown below.



Bracketing Interval Endpoints

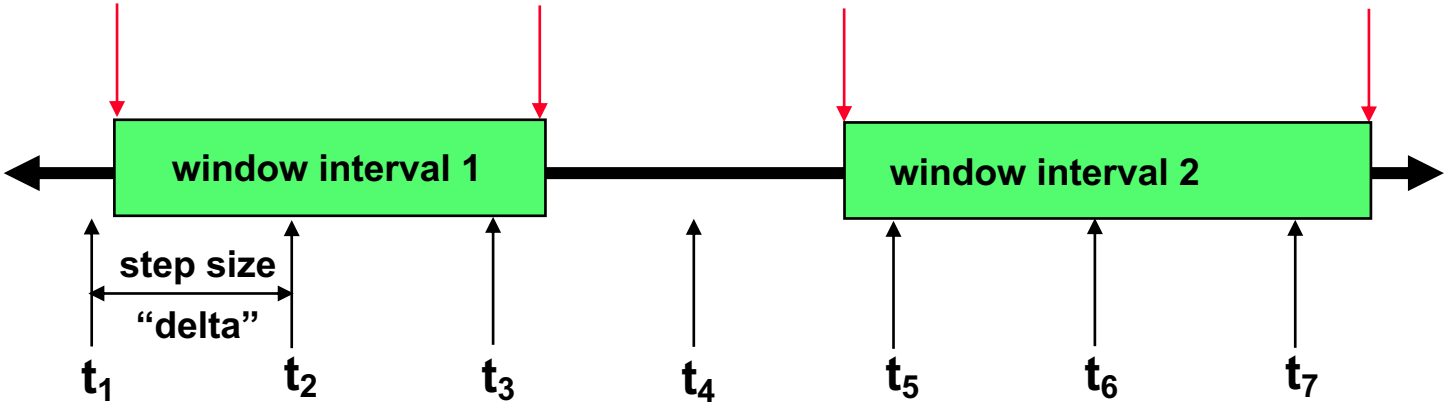
- In the diagram below, the green boxes denote intervals of a window.
- The start of the first interval is bracketed by the first and second times; the end of the first interval is bracketed by the third and fourth times. The start of the second interval is bracketed by the fourth and fifth times.



- **The step size is a critical determinant of the completeness of the solution:**
  - If the step size were equal to 3*delta, the first interval would not be seen.
  - If the step size were equal to 2*delta, the first and second intervals would not be seen as distinct.

Solving for Interval Endpoints

Navigation and Ancillary Information Facility



- Once an interval endpoint is bracketed, the GF subsystem performs a refinement search to locate the endpoint precisely (shown by the red arrows).
  - The default tolerance for convergence is 1.e-6 second.
- The refinement search is usually relatively fast compared to the interval bracketing search.

The Result Window

- For binary state searches, the window whose endpoints are found IS the result window.
  - The search is done once the endpoint refinement step has been completed for each interval over which the state is true.
- For numeric searches, once the monotone windows have been found, the result window still must be computed:
  - Local and absolute extrema can be found without further searching.
  - Equalities, inequalities, and adjusted absolute extrema require a second search pass in which each monotone interval is examined.
    - » These searches don't require sequential stepping and are usually relatively fast compared to the interval bracketing search.
- Since the roots are found by a search process, they are subject to approximation errors.
  - NOTE: The geometric condition may not be satisfied at or near the endpoints of the result window's intervals.
- Usually data errors are large enough so that the accuracy of the result is poorer than its precision.

Workspace

Specifying Workspace Dimensions

- GF numeric scalar searches can require relatively large amounts of memory to store intermediate results.
  - For Fortran Toolkits, user applications must declare a buffer of workspace windows.
  - For C, IDL, and MATLAB Toolkits, users need only specify the maximum number of workspace window intervals that are needed; these Toolkits will dynamically allocate the amount of memory required by the specified workspace window interval count.
- In most cases, users need not accurately compute the required amount of workspace; it's usually safe to specify a number much larger than the amount actually needed.
  - For example, if the result window is anticipated to contain 100 intervals, specifying a workspace window interval count of 10000 will very likely succeed.
  - See the GF Required Reading and the API documentation for details.

API Example: GFDIST

Solve for Distance Constraints

API Example: GFDIST

- **GFDIST finds times when a specified constraint on the distance between two ephemeris objects is met.**
  - **The distance is the norm of a position vector**
  - **The position vector is defined using a subset of the inputs accepted by SPKPOS:**
    - **Target**
    - **Observer**
    - **Aberration Correction**
  - **The constraint is a numeric relation: equality or inequality relative to a reference value, or attainment of a local or absolute extremum.**
- **The search is conducted within a specified confinement window.**
- **The search produces a result window which indicates the time period, within the confinement window, over which the constraint is met.**

Language Differences

- **Due to use of SPICE windows, some of the GFDIST set-up code differs substantially across languages.**
  - **We'll show how to perform the set-up unique to each language.**
    - » **Note: there's no set-up to do in the MATLAB case; hence there's no MATLAB-specific set-up slide.**
  - **The rest of the code is sufficiently parallel across languages to justify showing only the Fortran code.**
  - **Note however that the treatment of workspace differs across languages: only in Fortran does the user application have to pass the workspace array to the GF API routine.**

Fortran Set-up

Fortran constant and variable declarations

Declare confinement window, result window, and workspace array

```
INCLUDE 'gf.inc'
  ...
INTEGER                 LBCELL
PARAMETER               ( LBCELL =  -5 )
INTEGER                 MAXWIN
PARAMETER               ( MAXWIN = 200000 )
DOUBLE PRECISION        CNFINE  ( LBCELL : MAXWIN )
DOUBLE PRECISION        RESULT ( LBCELL : MAXWIN )
DOUBLE PRECISION        WORK    ( LBCELL : MAXWIN, NWDIST )
```

Include GF parameters such as NWDIST

Choose a "large" value for window size (called "MAXWIN" here), if you're not sure what size is required. The actual requirement depends on underlying geometry, confinement window, and search step size.

Initialization…typically done once per program execution

Initialize confinement and result windows. Workspace need not be initialized here.

```
CALL SSIZED ( MAXWIN, CNFINE )
CALL SSIZED ( MAXWIN, RESULT )
```

C Set-up

C constant and variable declarations

Declare confinement and result windows, as well as size of workspace.

```
#include "SpiceUsr.h"
  ...

#define  NINTVL        100000
#define  MAXWIN        ( 2 * NINTVL )

SPICEDOUBLE_CELL   ( cnfine,  MAXWIN );
SPICEDOUBLE_CELL   ( result,  MAXWIN );
```

Include CSPICE macro, typedef, and prototype declarations

Choose a "large" value for window size (called "MAXWIN" here), if you're not sure what size is required. Actual requirement depends on underlying geometry, confinement window, and search step size. The window size must be twice the maximum number of intervals the window is meant to hold.

These macro calls declare CSPICE window structures and set their maximum sizes.

IDL Set-up

IDL window creation

Icy windows are created dynamically:

cnfine = cspice_celld (MAXWIN)

Choose a "large" value for window size (called "MAXWIN" here), if you're not sure what size is required. Actual requirement depends on underlying geometry, confinement window, and search step size. The window size must be twice the maximum number of intervals the window is meant to hold.

The output result window is created by CSPICE_GFDIST; it does not require a constructor call by the user application.

All Languages: Additional Set-up

Navigation and Ancillary Information Facility

Initialization…typically done once per program execution

Tell your program which SPICE files to use ("loading" files)

```
CALL FURNSH  ('spk_file_name')
CALL FURNSH  ('leapseconds_file_name')
```

Better yet, replace these two calls with a single call to a "meta-kernel" containing the names of all kernel files to load.

The next step is to insert times into the confinement window. The simplest confinement window consists of a single time interval.

Convert UTC start and stop times to ephemeris time (TDB), if needed:

```
CALL STR2ET ( 'utc_start', tdb_0)
CALL STR2ET ( 'utc_stop', tdb_1)
```

Insert start and stop times into the confinement window:

```
CALL WNINSD ( tdb_0, tdb_1, CNFINE )
```

Execute the Search -1

Search execution…done as many times as necessary during program execution

Choose:

- Geometric input arguments:
  - » Target
  - » Observer
  - » Aberration correction
- Constraint arguments:
  - » Relation
  - » Reference value, if applicable
  - » Adjustment value, if applicable
- Search step size

Then call GFDIST:

CALL GFDIST (target, 'correction', observer, 'relate',
refval, adjust, step, cnfine, mw, nw, *work,* ***result*** **)**

inputs | Input/output | output

Execute the Search -2

Search execution, continued

Extract intervals from the result window:

```
DO I = 1, WNCARD( RESULT )

   [Fetch the endpoints of the Ith interval of the result window.]
   CALL WNFETD( RESULT, I, START, FINISH )
         [ use START, FINISH… ]
END DO
```

- **Note:**
  - The result window may be empty.
  - The constraint might not be satisfied at or near the endpoints of any interval of RESULT.
    - » Consider using the window contraction routine WNCOND to shrink the intervals of the result window slightly, so the constraint is met on the entire result window.
    - » Caution: DON'T use WNCOND for minimum, maximum, or equality searches---the result window will disappear! (WNCOND is ok for adjusted absolute extrema search results, though, since the result intervals are not singletons.)
    - » Caution: using WNCOND may not be desirable if the window is an intermediate result: subsequent, derived results might be made less accurate.

Arguments of GFDIST - 1

INPUTS

- **TARGET* and OBSERVER*: Character names or NAIF IDs for the end point and origin of the position vector (Cartesian position and velocity vectors) whose length is the subject of the search. Below, we'll simply call this length "the distance."**
  - **The position vector points from observer to target.**

- **CORRECTION: Specification of what kind of aberration correction(s), if any, to apply in computing the distance.**
  - **Use 'LT+S' to use the apparent position of the target as seen by the observer. 'LT+S' invokes light time and stellar aberration corrections.**
  - **Use 'NONE' to use the uncorrected (aka "geometric") position, as given by the source SPK file or files.**

See the headers of the subroutines GFDIST and SPKEZR, the document SPK Required Reading, or the "Fundamental Concepts" tutorial for details. See the SPK tutorial backup charts for examples of aberration correction magnitudes.

*** Character names work for the target and observer inputs only if built into SPICE or if registered using the SPICE ID-body name mapping facility. Otherwise use the SPICE numeric ID in quotes, as a character string.**

Arguments of GFDIST - 2

INPUTS

- RELATE, REFVAL, ADJUST: parameters specifying a constraint to be met by the distance.
  - RELATE may be any of '=', '>', '<', 'LOCMAX', 'LOCMIN', 'ABSMAX', 'ABSMIN'
  - If RELATE is an equality or inequality operator, REFVAL is the corresponding double precision reference value. Units are km.
    - » For example, if the constraint is "distance = 4.D5 km," then RELATE is '=' and REFVAL is 4.D5.
  - If RELATE specifies an absolute maximum or minimum, ADJUST is the adjustment value. Units are km.
    - » Set ADJUST to 0.D0 for a simple absolute maximum or minimum.
    - » Set ADJUST to a positive value ADJ for either DISTANCE > absolute max - ADJ or DISTANCE < absolute min + ADJ.
- STEP: search step size, expressed as TDB seconds.
- CNFINE: the confinement window over which the search will be performed.
- MW, NW, WORK: the maximum capacity of each workspace window, the number of workspace windows, and the workspace array.

OUTPUTS

- RESULT: the window of times over which the distance constraint is satisfied.